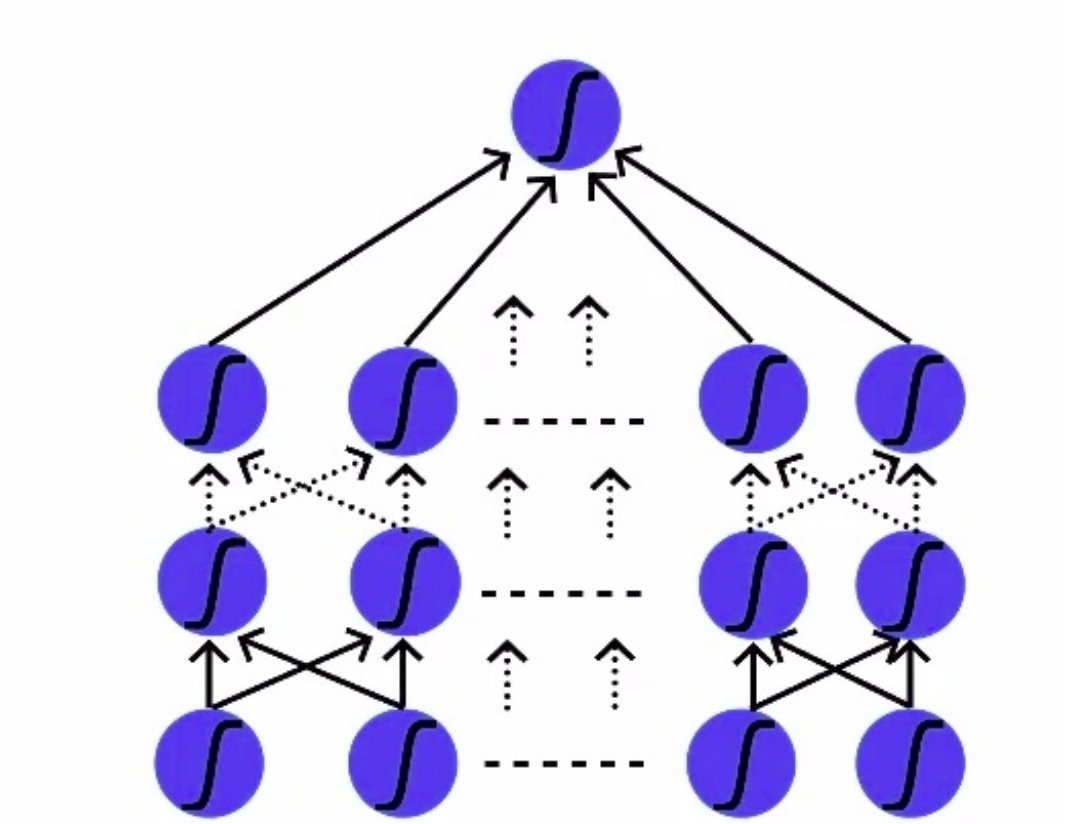
Notes on Feed-forward Neural Network

medium.com/@manveetdn/notes-on-feed-forward-neural-network-7537b3b59fbe

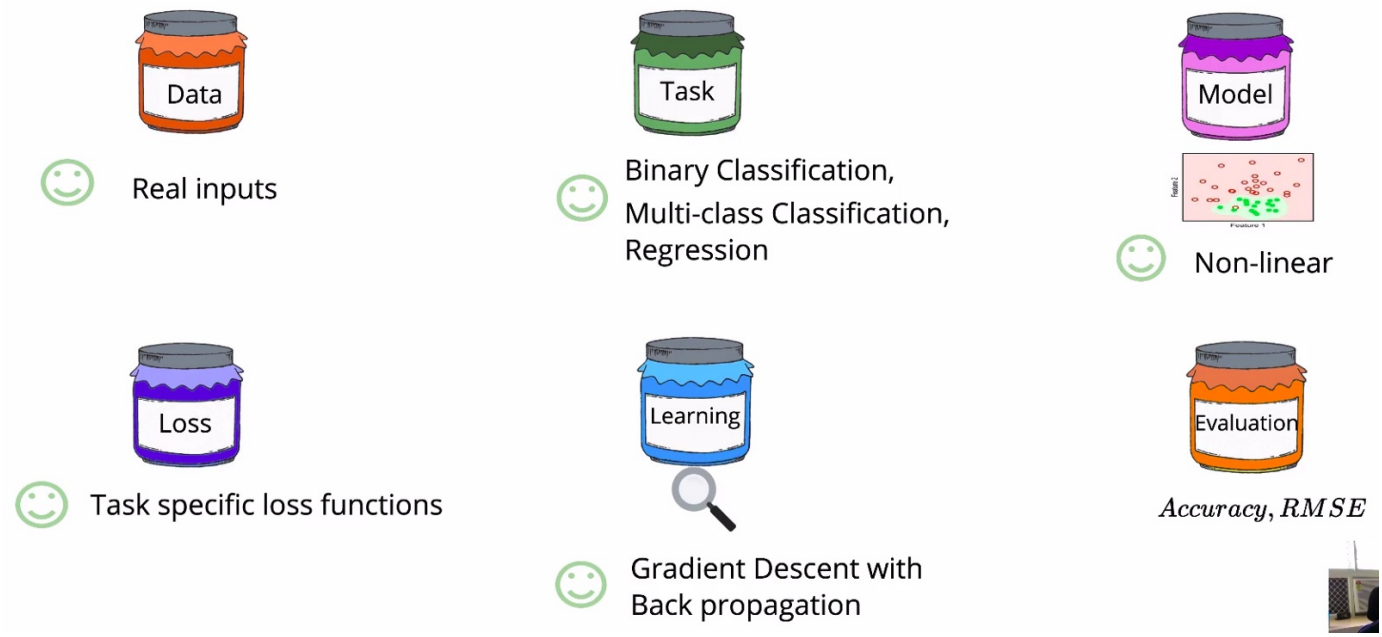
Deep Neural Network...

Disclaimer: This is notes on "Notes on Feed-forward Neural Network" Lesson (PadhAI onefourthlabs course "A First Course on Deep Learning")



(Source:OneFourthLabs)

This is a deep Neural network, We actually use this when we have a non linearly separable data-set. In this case we initiate with one sigmoid neuron, with help of that, we will we place many sigmoid neural adjacent to that and to top of that,to get a desired network that outputs required value, That deep neurons we will finally come out with a complex function that will predicts the values with

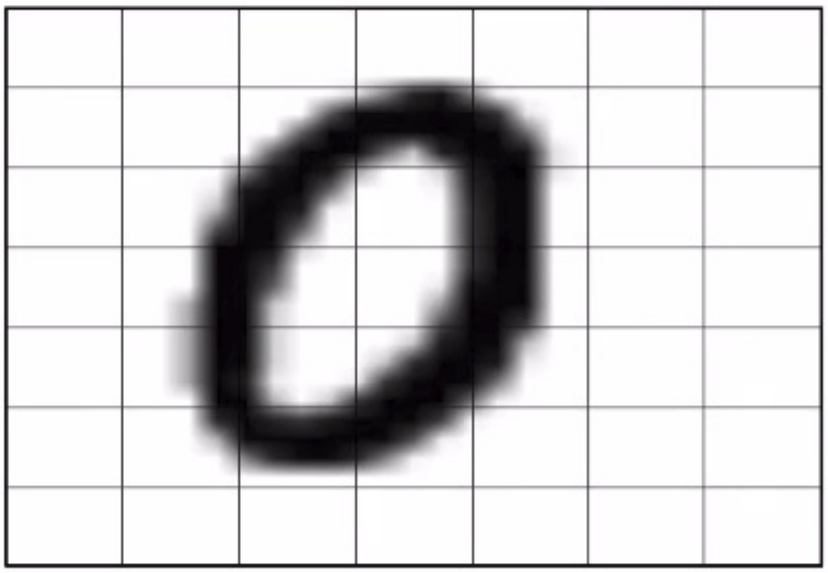


(Source:OneFourthLabs)

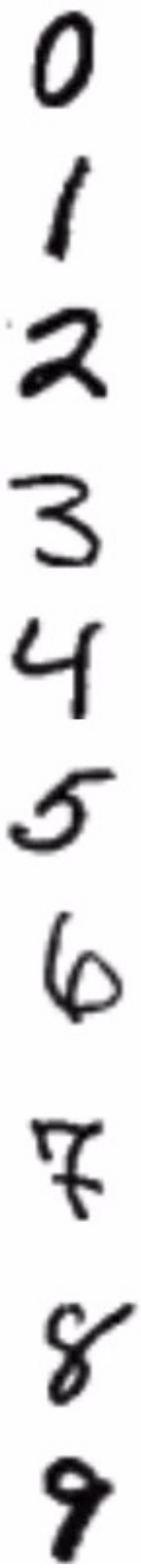
Here are we going to take **real inputs** in data jar , In the task section ,we are going to take the **Multi-class Classification** along with Binary Classification and regression.Here, we are going to deal with **non linearly separable** data-set, Loss function is specified based on the task and in the leaning jar we are going to deal with **Gradient Descent and Back Propagation** and in the Evaluation we are going to see **normal accuracy and Root mean square Error formulae**.

Lets see these all jars in case of the deep neural network

Data and Task:



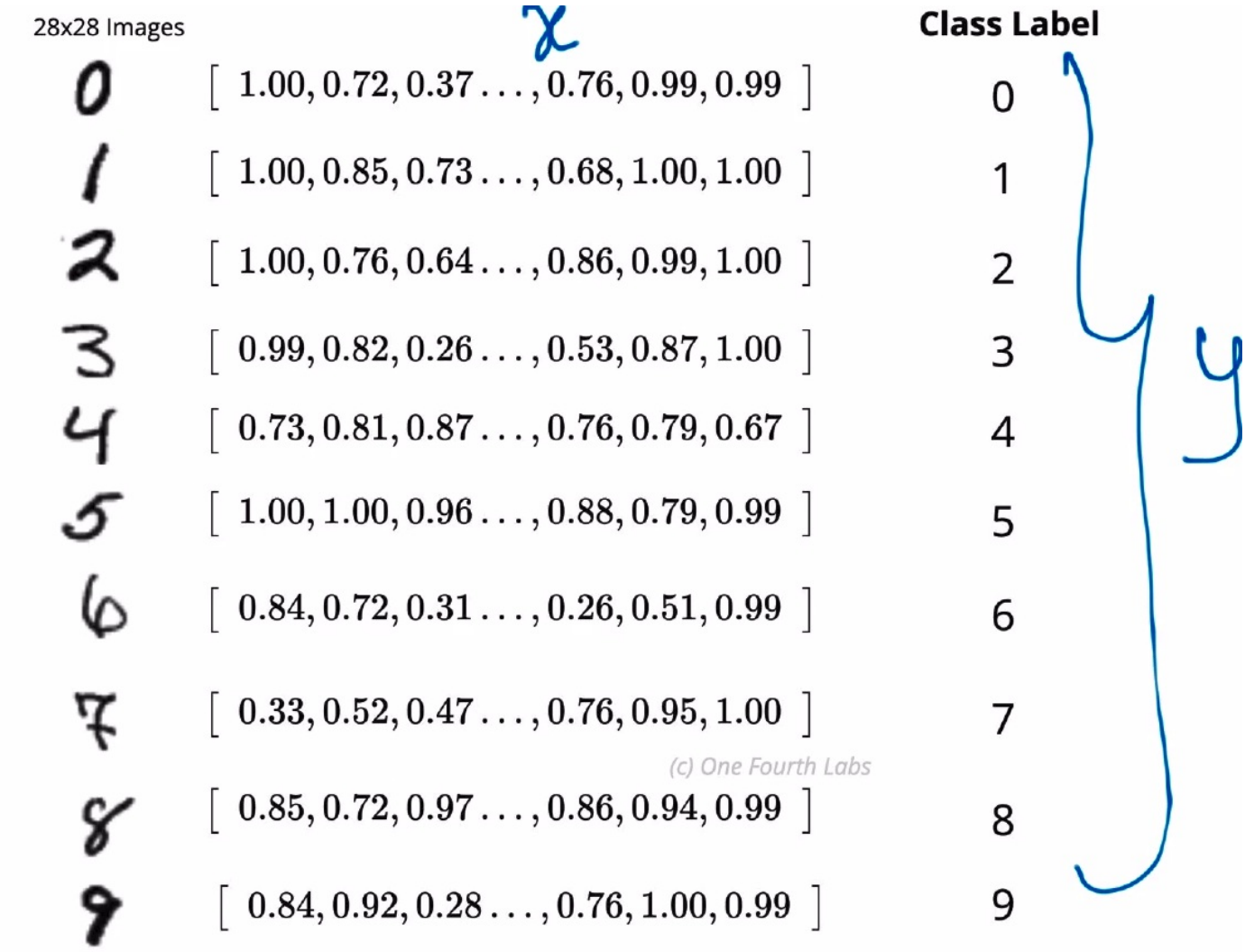
28x28 Images



How can we represent MNIST images as a vector ?

- Using pixel values of each cell
- Matrix having pixel values will be of size 28x28 (As MNIST images are of size 28x28)
- Each pixel value can range from 0 to 255. Standardise pixel values by dividing with 255
- Now, Flatten the matrix to convert into a vector of size 784 (28x28)

MNIST data-set (Source: OneFourthLabs)



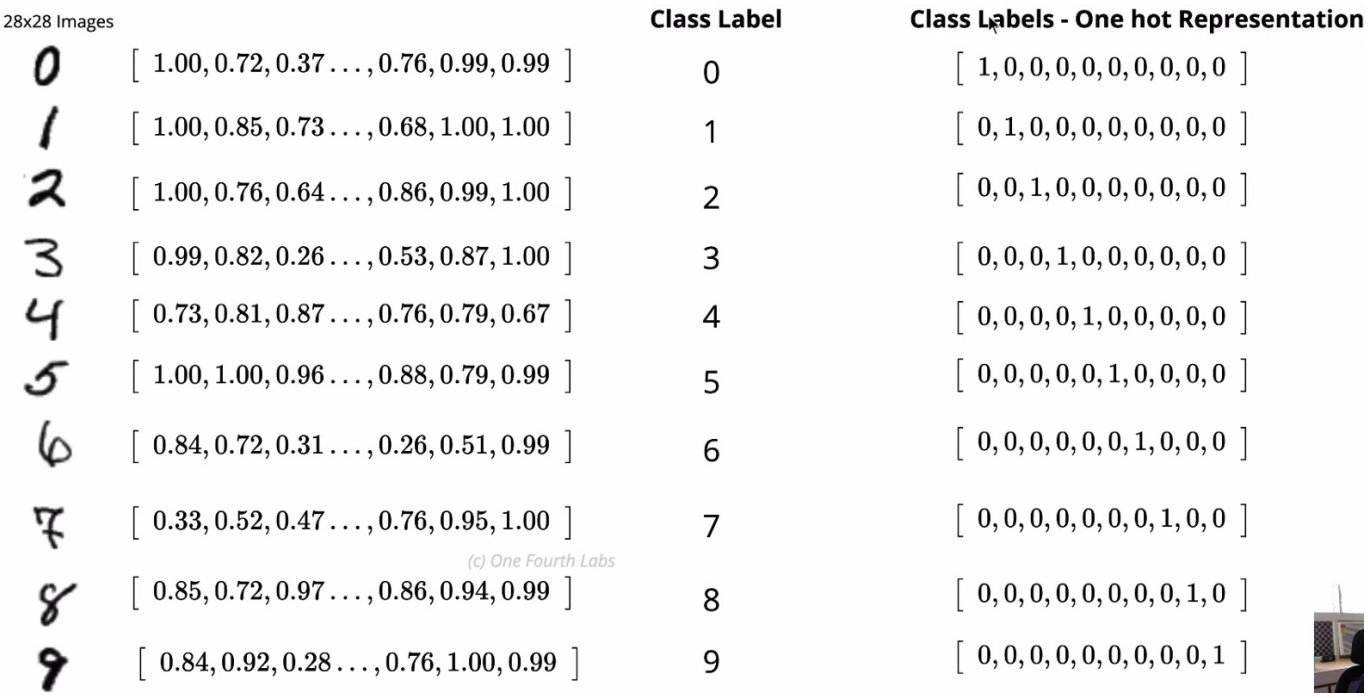
(Source:OneFourthLabs)

Here we are going to take a well know and familiar dataset called the **MNIST dataset** .Here each image is of the size (28X28) pixels and each pixel will have a value of its own from 0–255.

Now our task is that we are going to take all values of the pixels into a matrix from for each digit and we will take it as ‘X’ we also need a ‘Y’. Here ‘Y’ will be the the one of the label between 0 to 9 for each one matrix. This is called a **Multi-class Classification problem**.

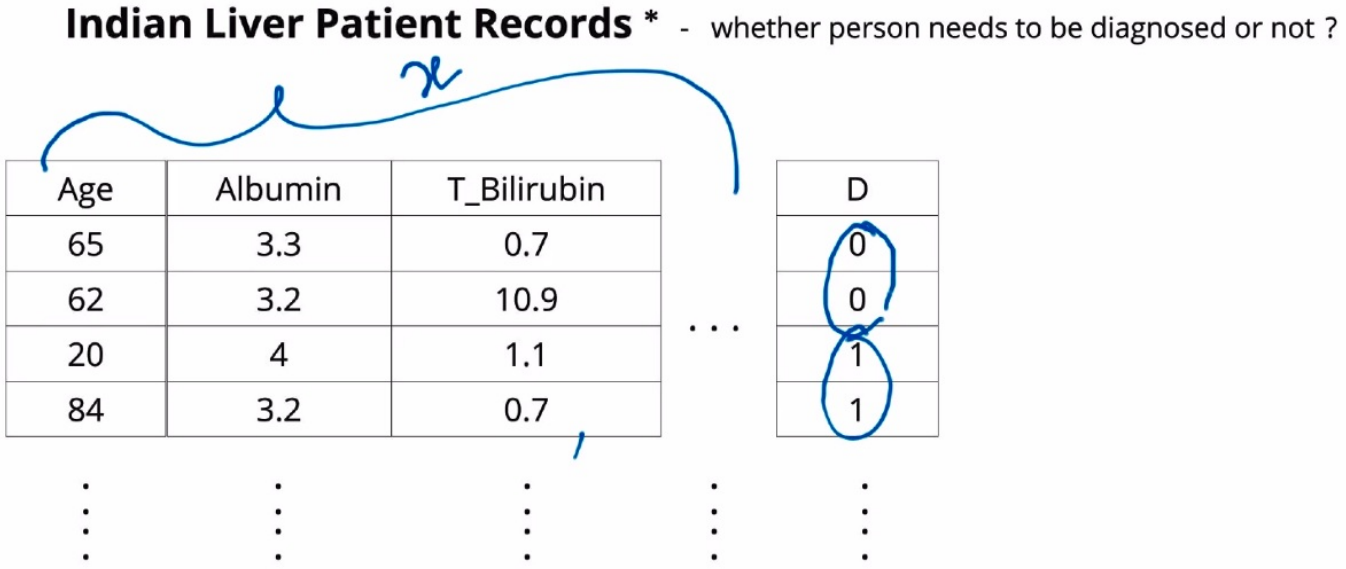
Multi-class Classification means we will have multiple class to choose the output from here in this case we have the many class i.e., from 0–9 where the output lies in any one .

Here we ca represent the class labels as the **One-hot representation** this means that a random variable Y which can take one of the ten values (0–9) and in particular image (Ex: 0) then the all the probability mass is concentrated on first label making it one all remaining all are made zeroes zero if the image is 1 then the second mass is concentrated more makings its position value one and remaining all zeroes.



One-hot Representation of each number (Source:OneFourthLabs).

Again we can also, take many data-sets like:



(Source:OneFourthLabs)

(i)Data-set is a needs binary classification to predict whether patient need to be diagnosed or not based on various parameters

Boston Housing* - Predict Housing Values in Suburbs of Boston

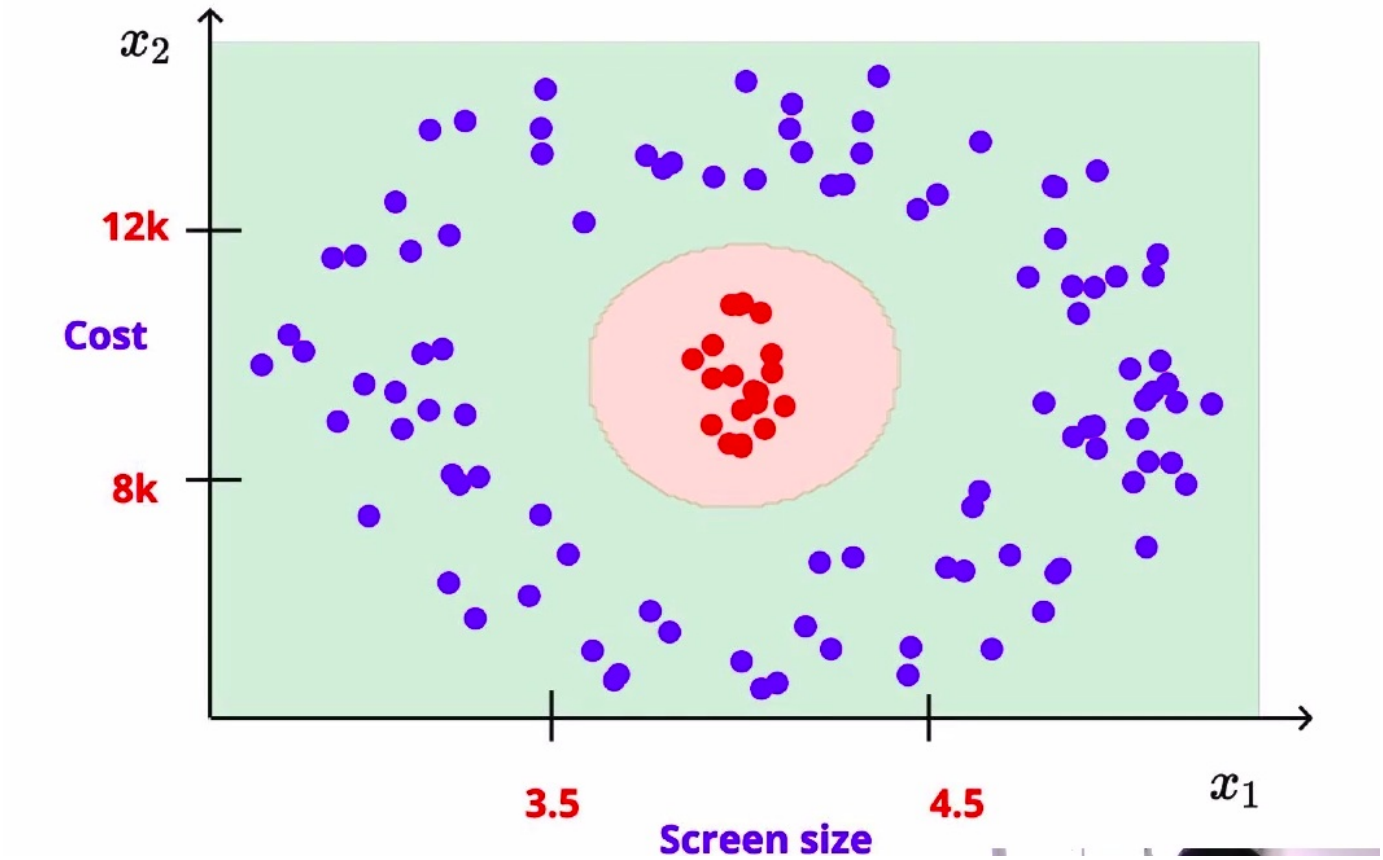
Crime	Avg No of rooms	Age		House Value
0.00632	6.575	65.2		24
0.02731	6.421	78.9	. . .	21.6
0.3237	6.998	45.8		33.4
0.6905	7.147	54.2		36.2
⋮	⋮	⋮	⋮	⋮

(Source:OneFourthLabs)

(ii) Data-set is a needs regression type of classification that its its takes various inputs of particular locality and we need to predict the value of the house

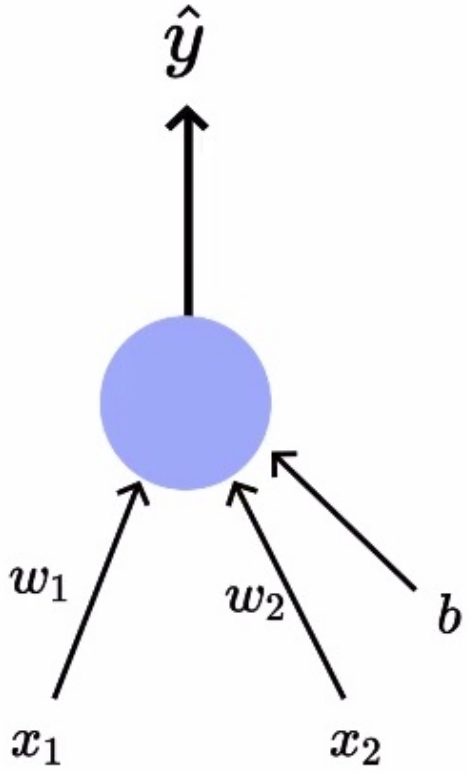
Model:

Here we will see how to built complex functions with deep neural network.



Mobile phones data-set (Source:OneFourthLabs)

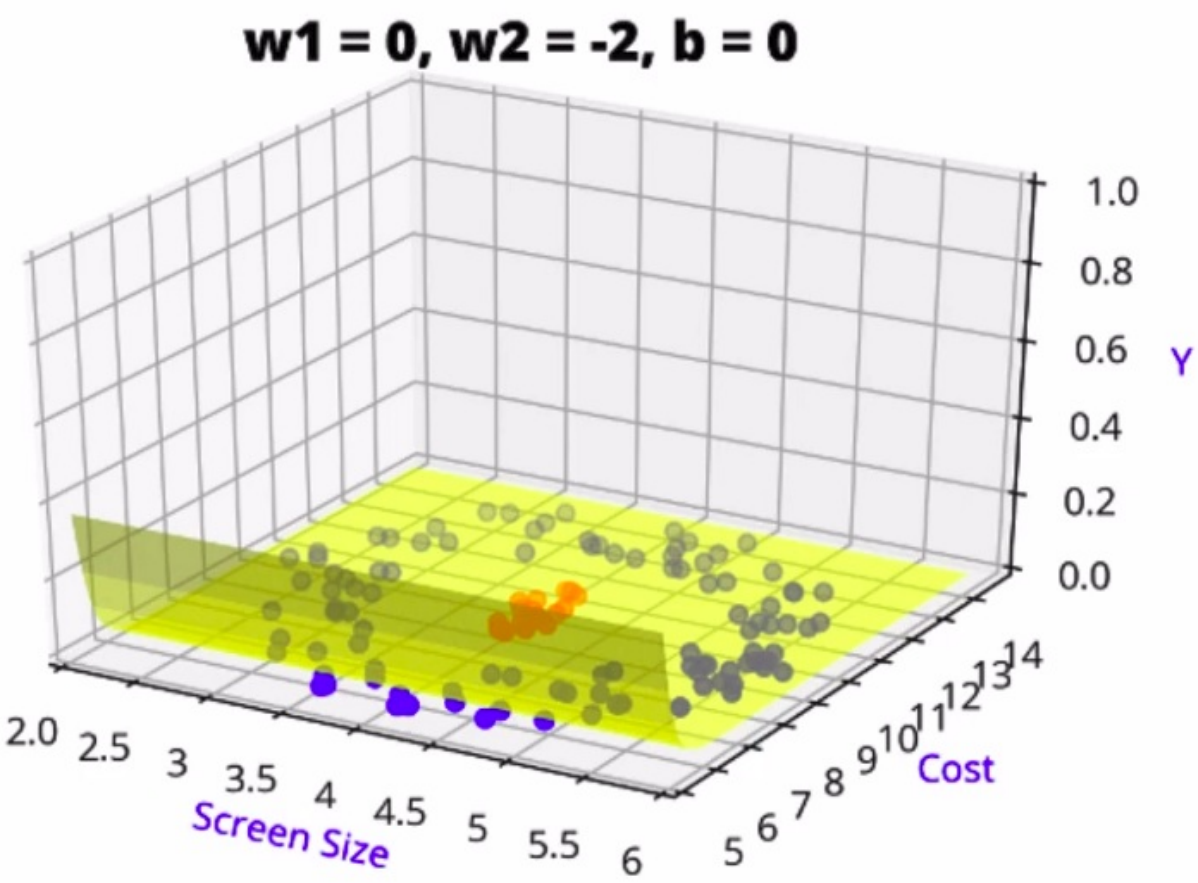
Lets take the data-set of mobile phones. Graph plotted between the Screen-size and the Cost of the mobile phone.the red points are the most selling phones and highly rated phones and we need to function classify red points and the blue points.

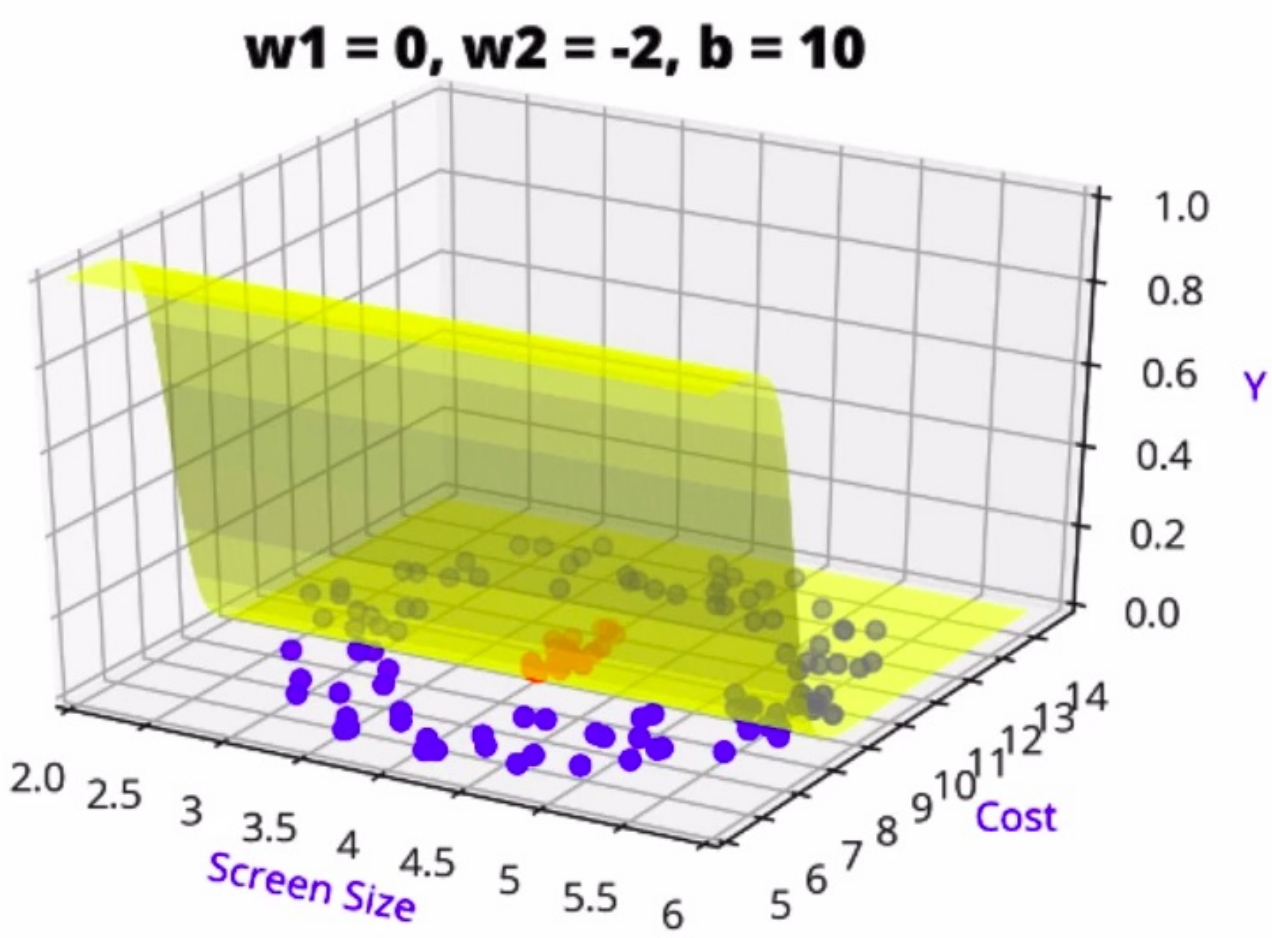


$$\hat{y} = f(x_1, x_2)$$

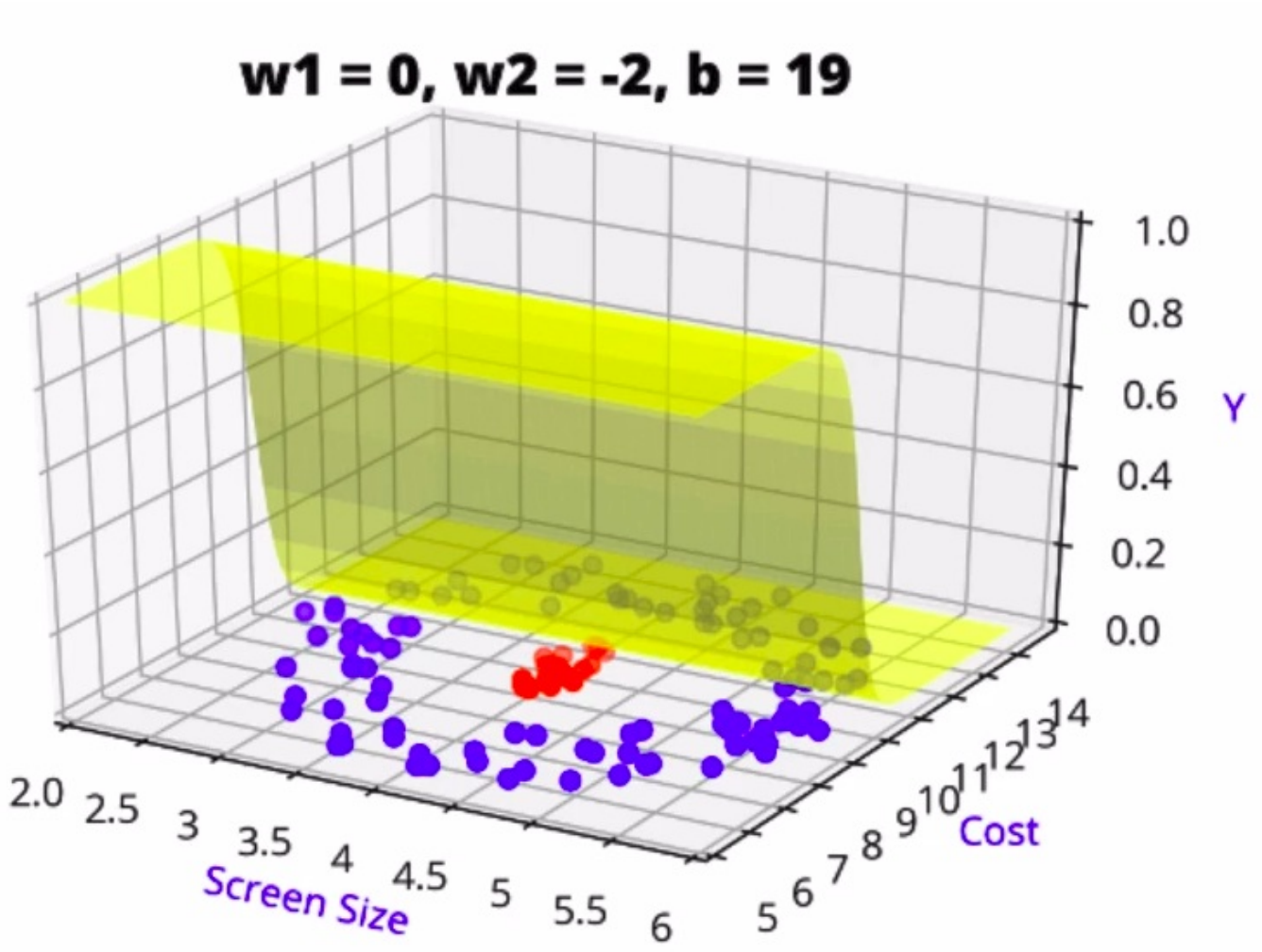
$$\hat{y} = \frac{1}{1+e^{-(w_1 * x_1 + w_2 * x_2 + b)}}$$

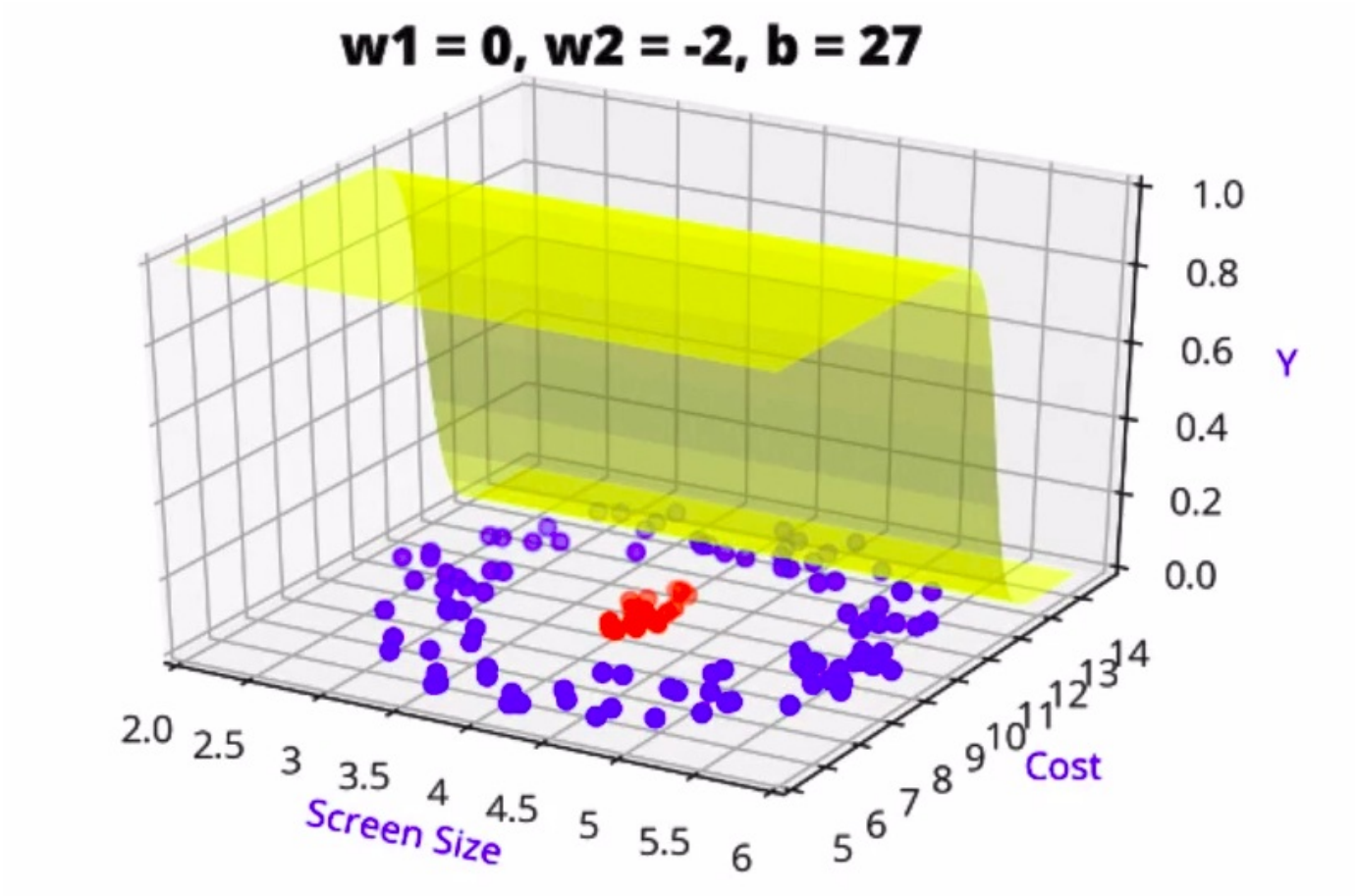
w_{11}


w1 = 0, w2 = -2, b = 0
1.0
0.8
0.6
0.4
0.2
0.0
Y
2.0
2.5
3
3.5
4
4.5
5
5.5
6
Screen Size
5
6
7
8
9
10
11
12
13
14
Cost

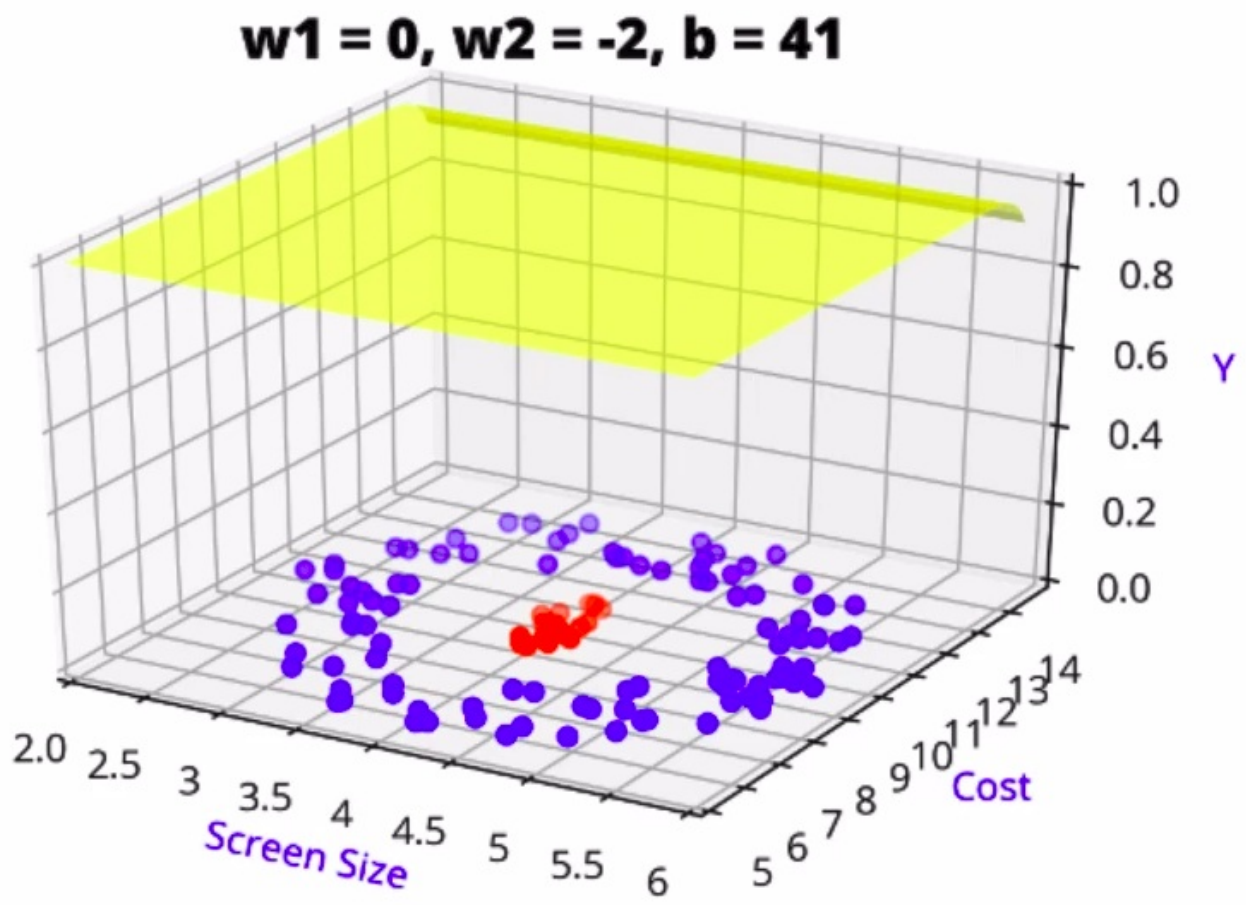


If we take the sigmoid function and how ever you manage its will not satisfy classifying red and blue points correctly.


w1 = 0, w2 = -2, b = 19
1.0
0.8
0.6
0.4
0.2
0.0
Y
2.0
2.5
3
3.5
4
4.5
5
5.5
6
Screen Size
5
6
7
8
9
10
11
12
13
14
Cost



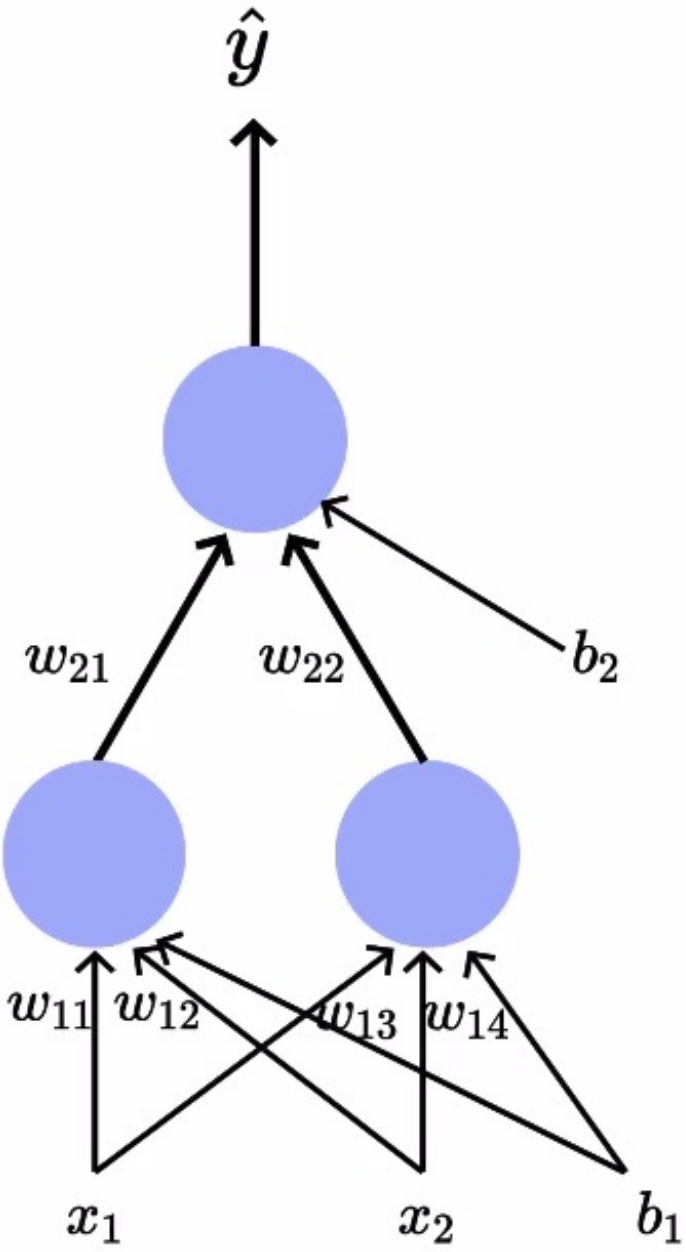
(Source:OneFourthLabs)



(Source:OneFourthLabs)

At some point while you keep on updating the values at last the sigmoid function turns up like this not classifying any point,ending up like this.

Now we will start building our deep neural network.


$\hat{y}$
w_{21}
w_{22}
b_2
w_{11}
w_{12}
w_{13}
w_{14}
x_1
x_2
b_1

(Source:OneFourthLabs)

$$h_1 = f_1(x_1, x_2)$$

$$h_2 = f_2(x_1, x_2)$$

$$\hat{y} = g(h_1, h_2)$$

$$h_1 = \frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}}$$

$$h_2 = \frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}}$$

$$\hat{y} = \frac{1}{1+e^{-(w_{21}*h_1+w_{22}*h_2+b_2)}}$$

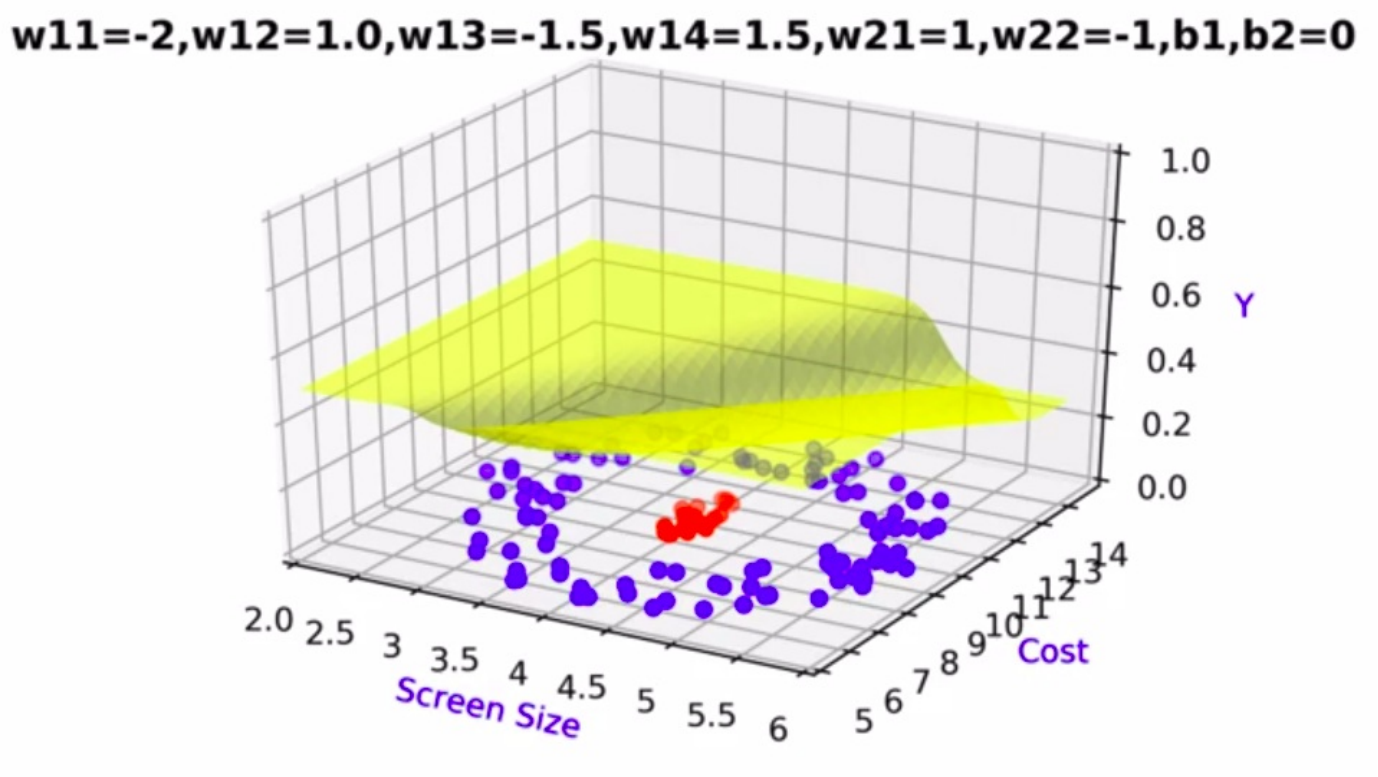
$$\hat{y} = \frac{1}{1+e^{-(w_{21}*h_1+w_{22}*h_2+b_2)}}$$

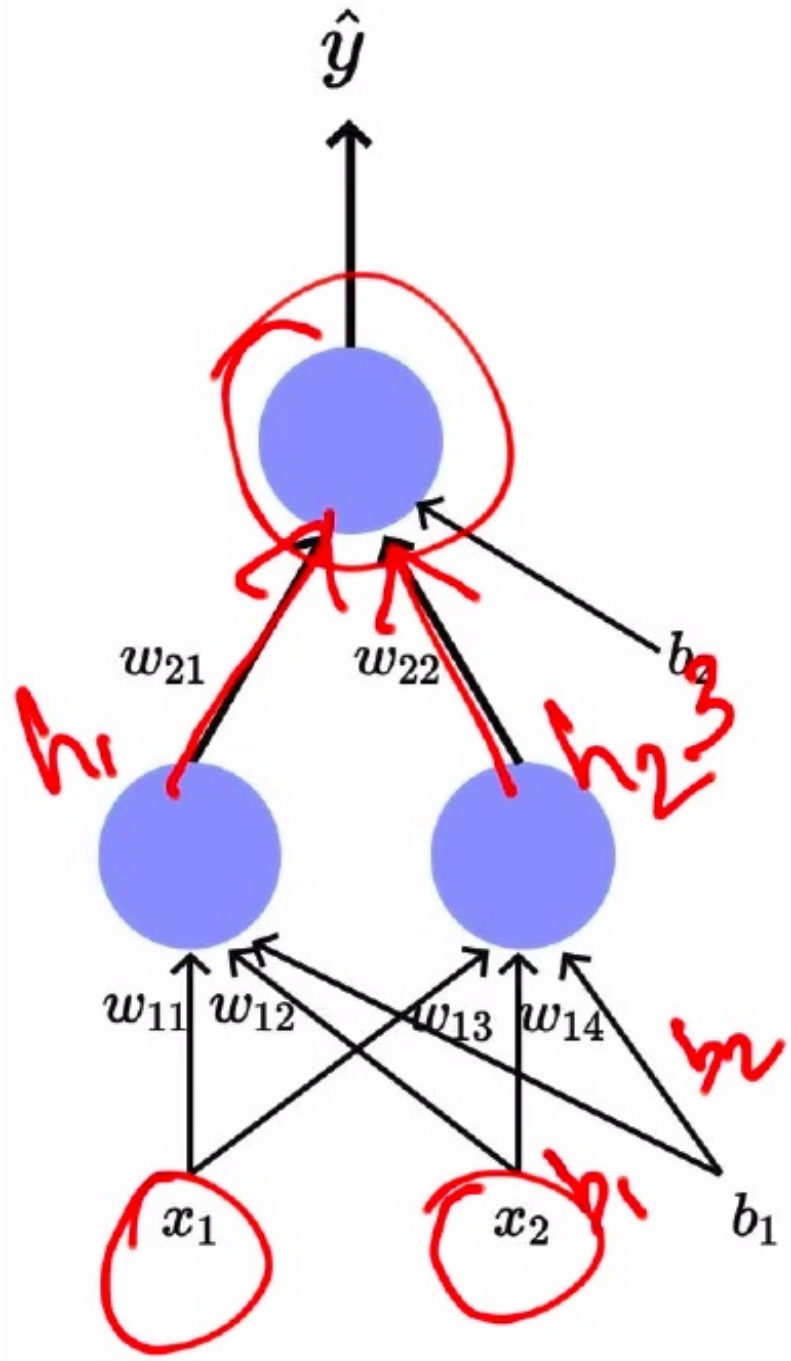
3.5

Screer

$$= \frac{1}{1+e^{-(w_{21}*(\frac{1}{1+e^{-(w_{11}*x_1+w_{12}*x_2+b_1)}})+w_{22}*(\frac{1}{1+e^{-(w_{13}*x_1+w_{14}*x_2+b_1)}})+b_2)}}$$

(Source:OneFourthLabs)


w11=-2,w12=1.0,w13=-1.5,w14=1.5,w21=1,w22=-1,b1,b2=0
1.0
0.8
0.6
0.4
0.2
0.0
Y
2.0
2.5
3
3.5
4
4.5
5
5.5
6
Screen Size
5
6
7
8
9
10
11
12
13
14
Cost



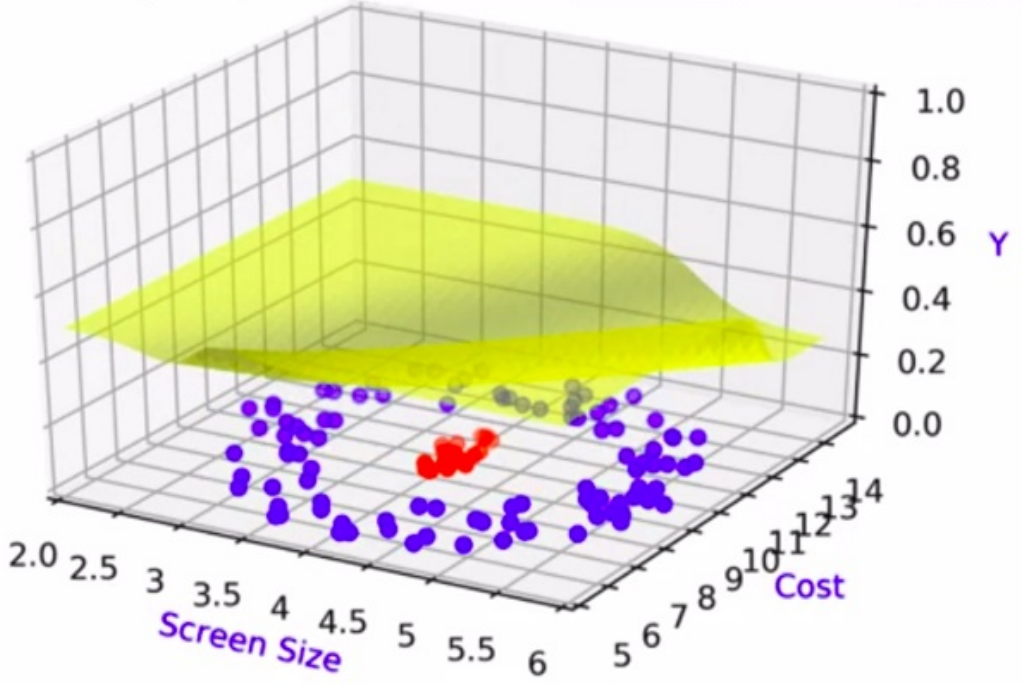
Here with the deep neural network ,the function stratifies most of the points but not all the points, but its better than before case.(Source:OneFourthLabs)

This is a basic deep neural network with h1 and h2 and the function “y hat” as the complex function built from these two functions h1 and h2.

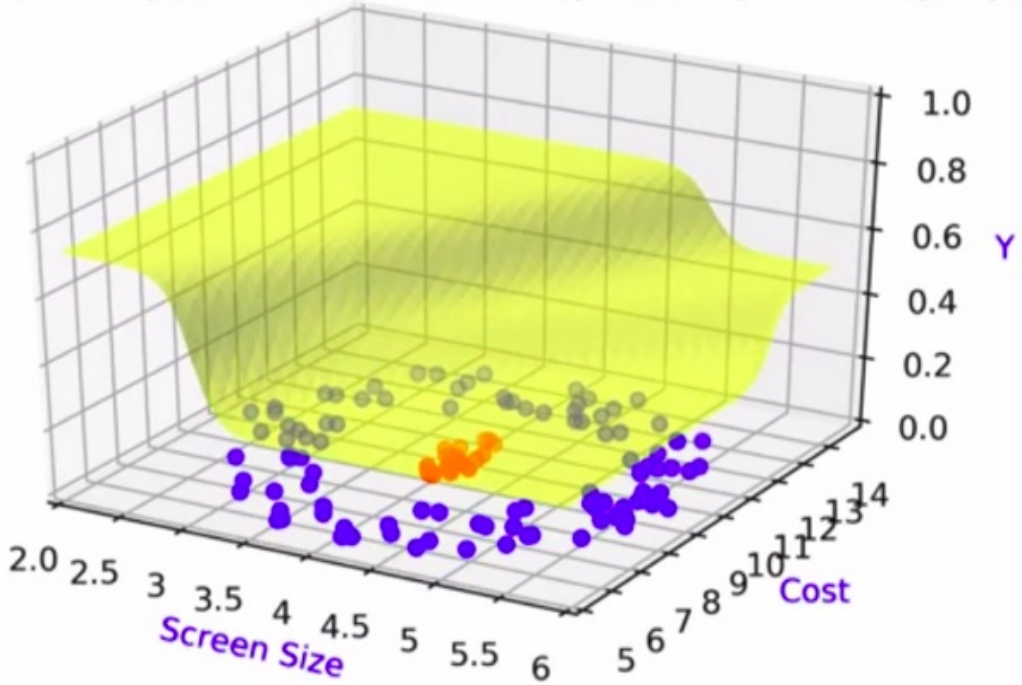
We get the final equation by substituting h1 and h2 values in the function.

like that updating the weights we can come out with different function which classify even more better than the before like below.

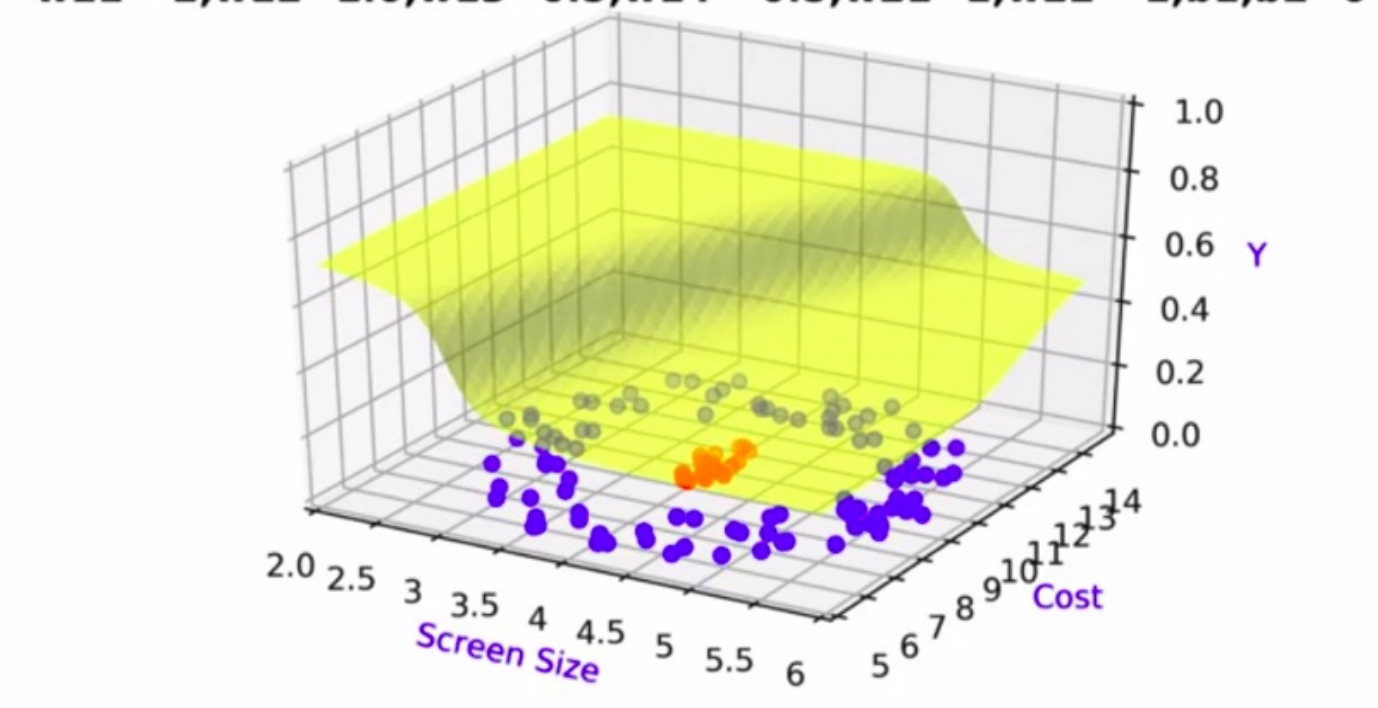
w11=-1,w12=0.5,w13=-2.0,w14=2.0,w21=1,w22=-1,b1,b2=0



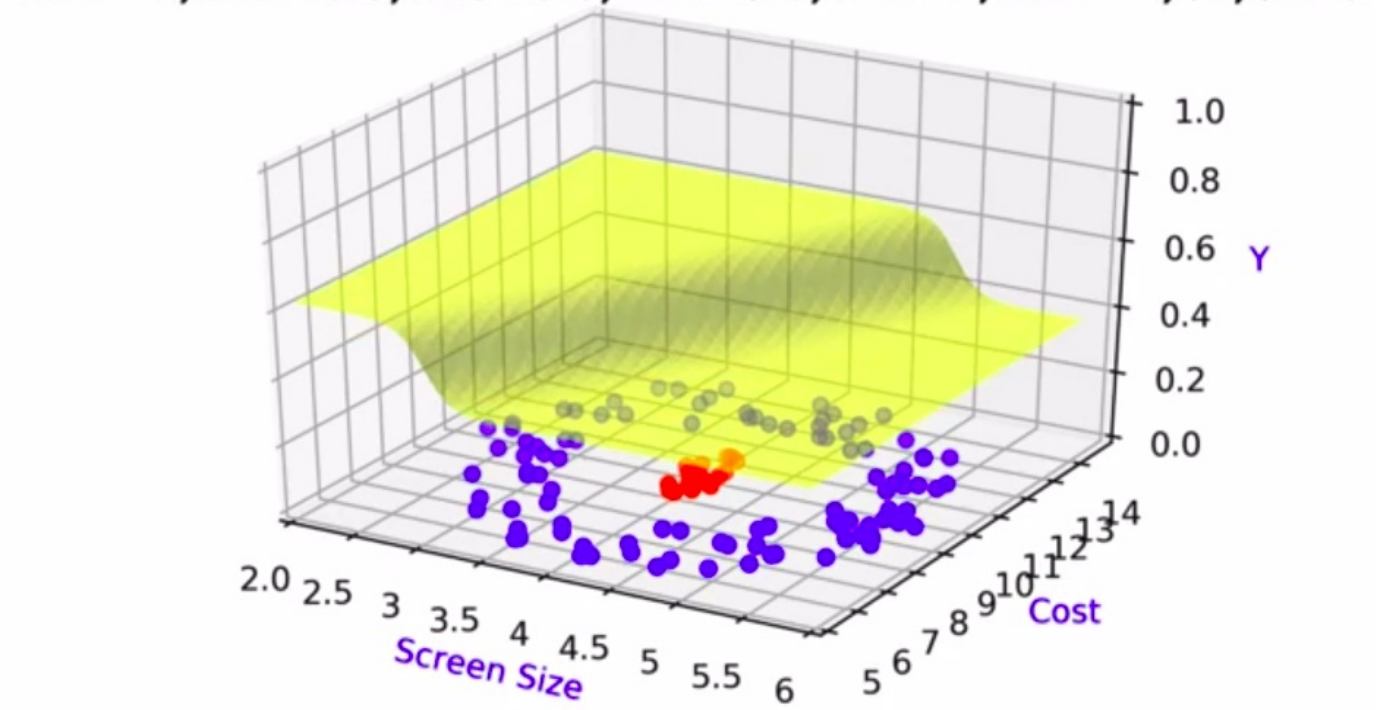
w11=-2,w12=1.0,w13=2.0,w14=-2.0,w21=1,w22=-1,b1,b2=0

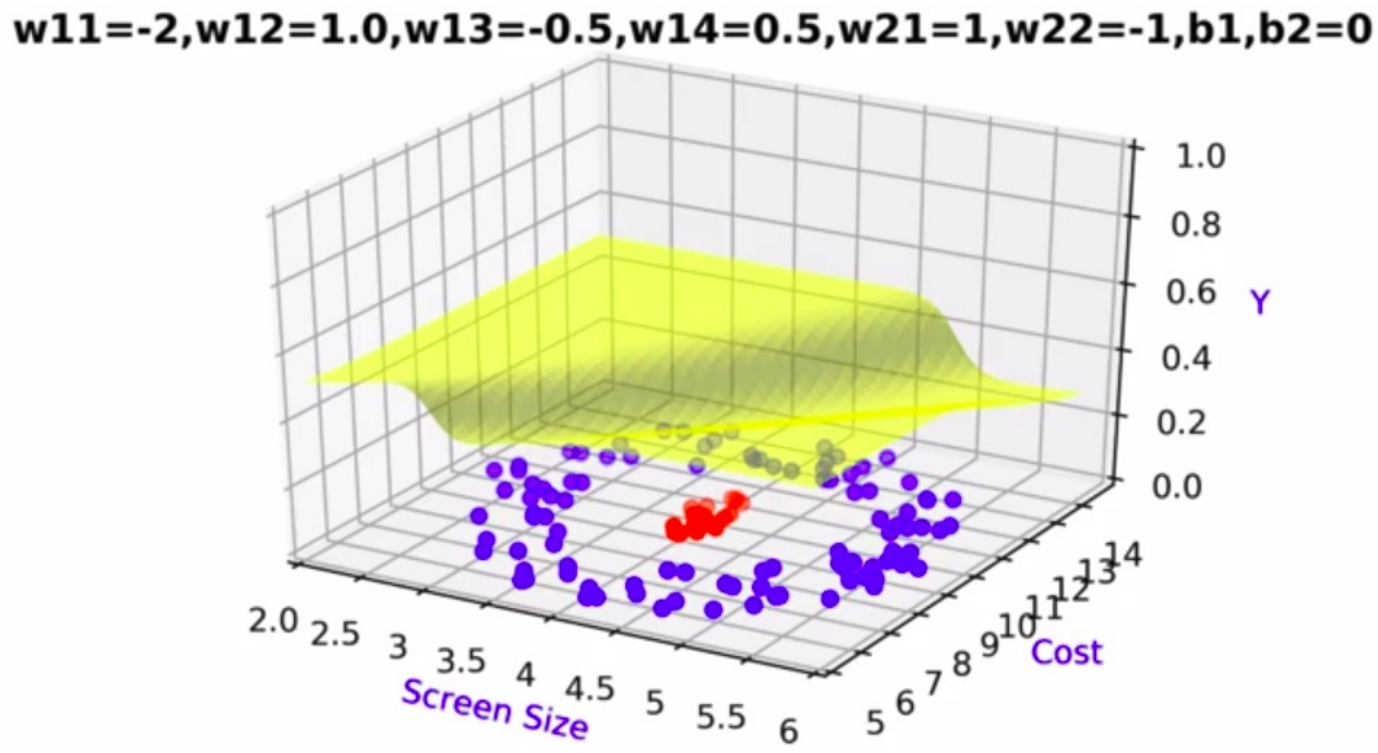


w11=-2,w12=1.0,w13=0.5,w14=-0.5,w21=1,w22=-1,b1,b2=0

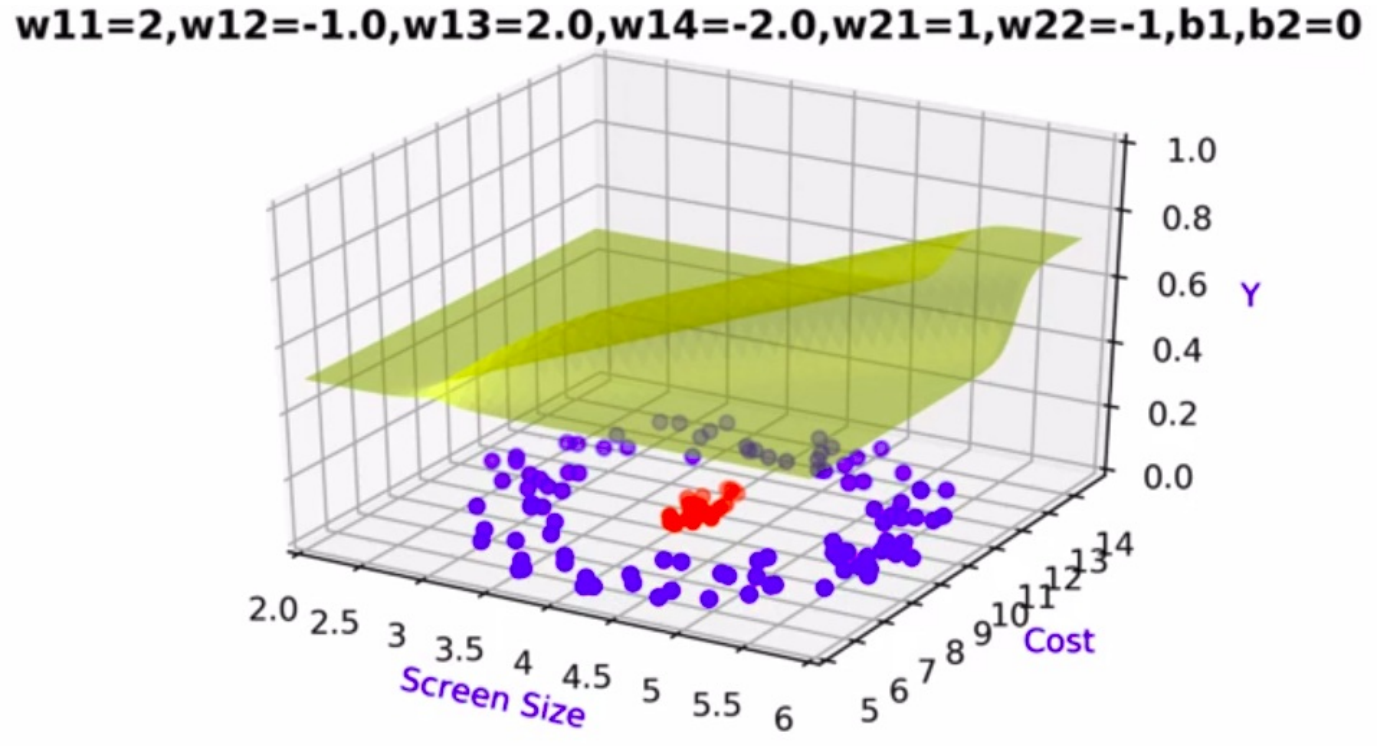


w11=-2,w12=1.0,w13=0.0,w14=0.0,w21=1,w22=-1,b1,b2=0





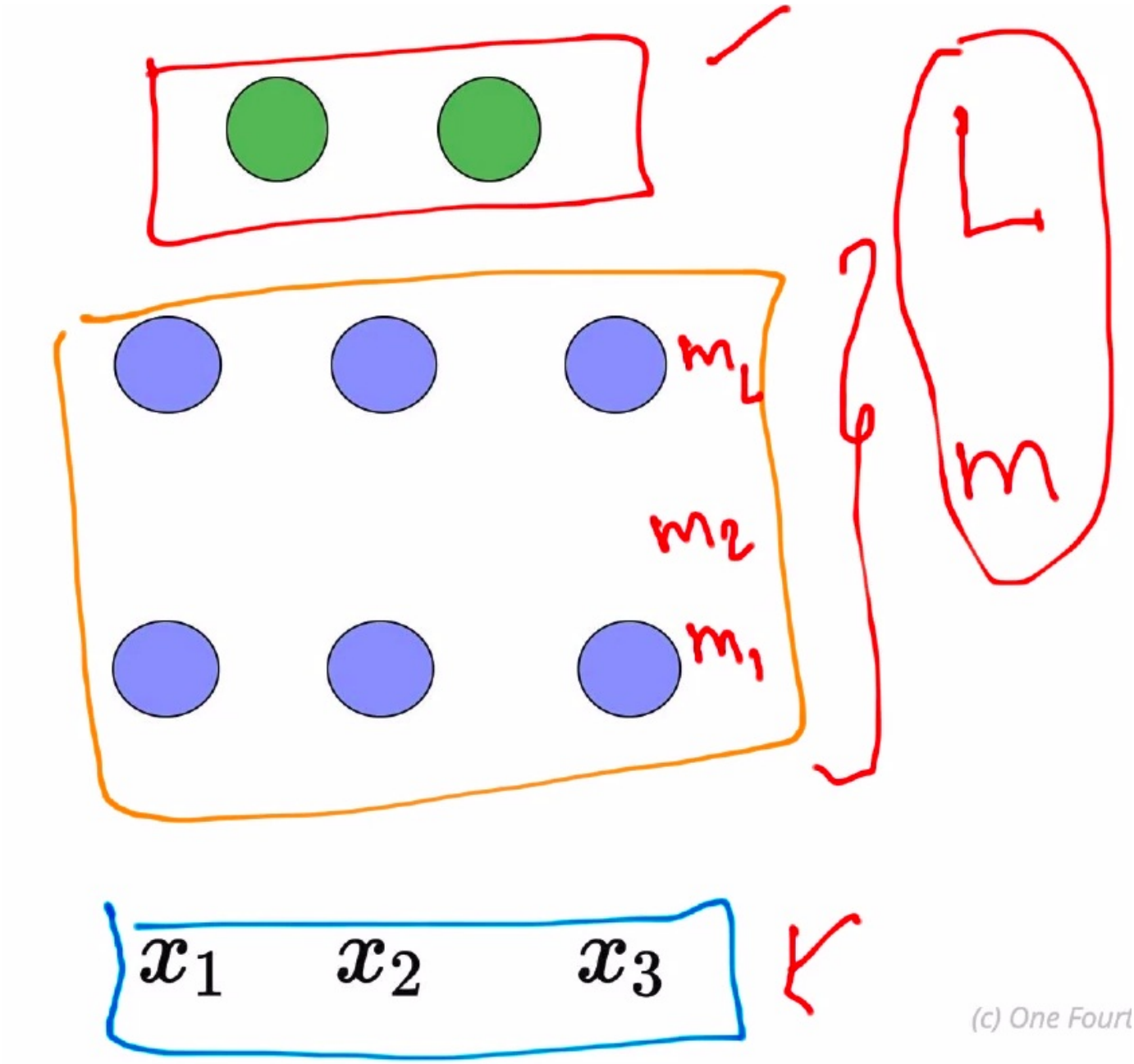
Updating weights helps us increase the accuracy bringing out the better model. (Source:OneFourthLabs)



This is the final one which classifies all the points correctly.(Source: OneFourthLabs)

Finally for a particular value of weights , will get a perfect model with classifies all the points correctly like **red points as 1 and blue points as 0.**

> So here we proved the universal approximation theorem which says " **if we have a deep neural network we can approximate arbitrary functions.**"

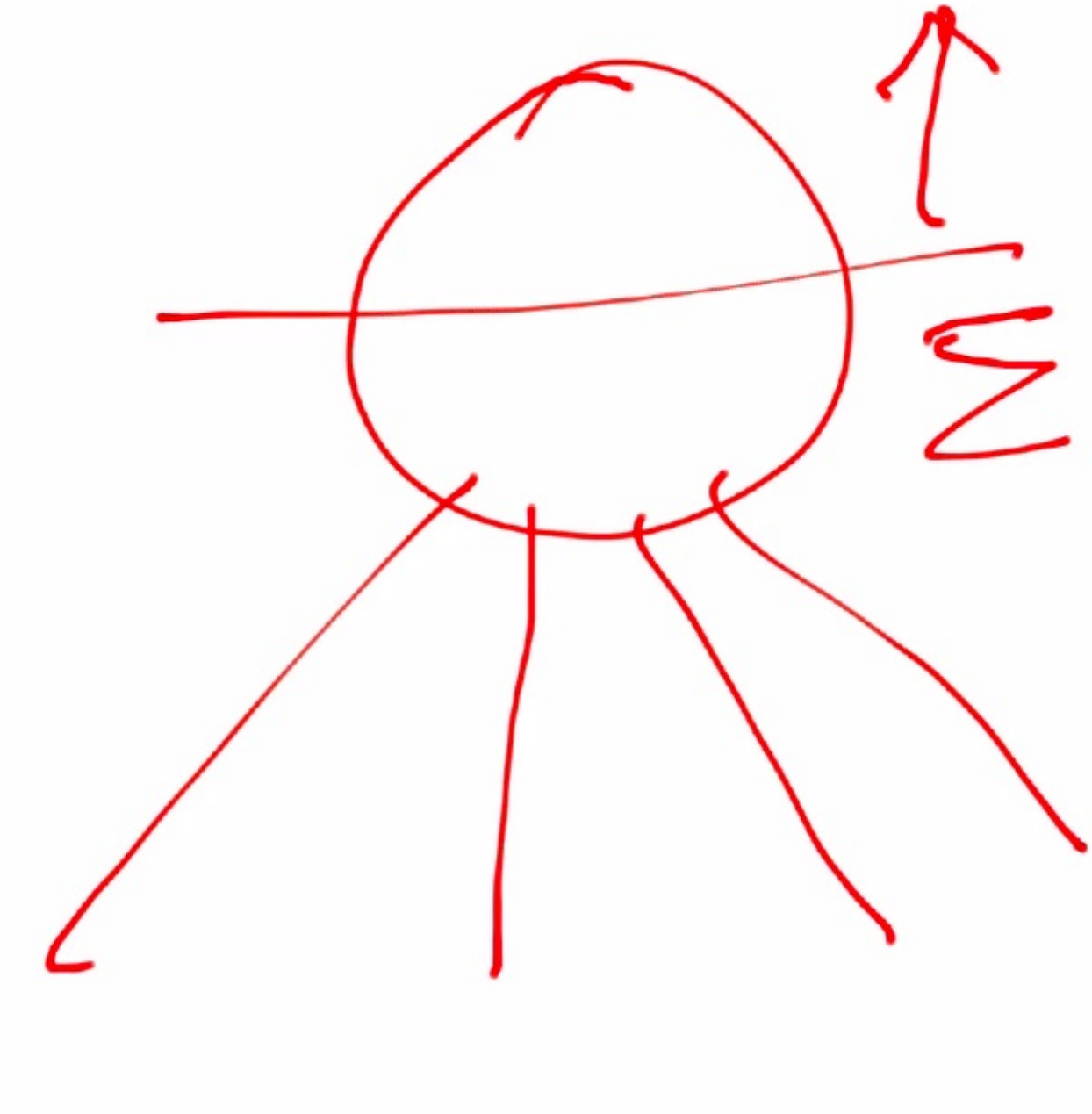


(Source:OneFourthLabs)

Till here this s specific configuration like we have 2 neurons in the first layer and one neuron in the second layer, If we go in more generic way

If you are given an 'n' dimensional input,then we will have on layer of neurons and other layer of neurons on that, and many more layers like that, one above the other.

Lets consider about a "L" layered network containing "m" neurons in each layer.

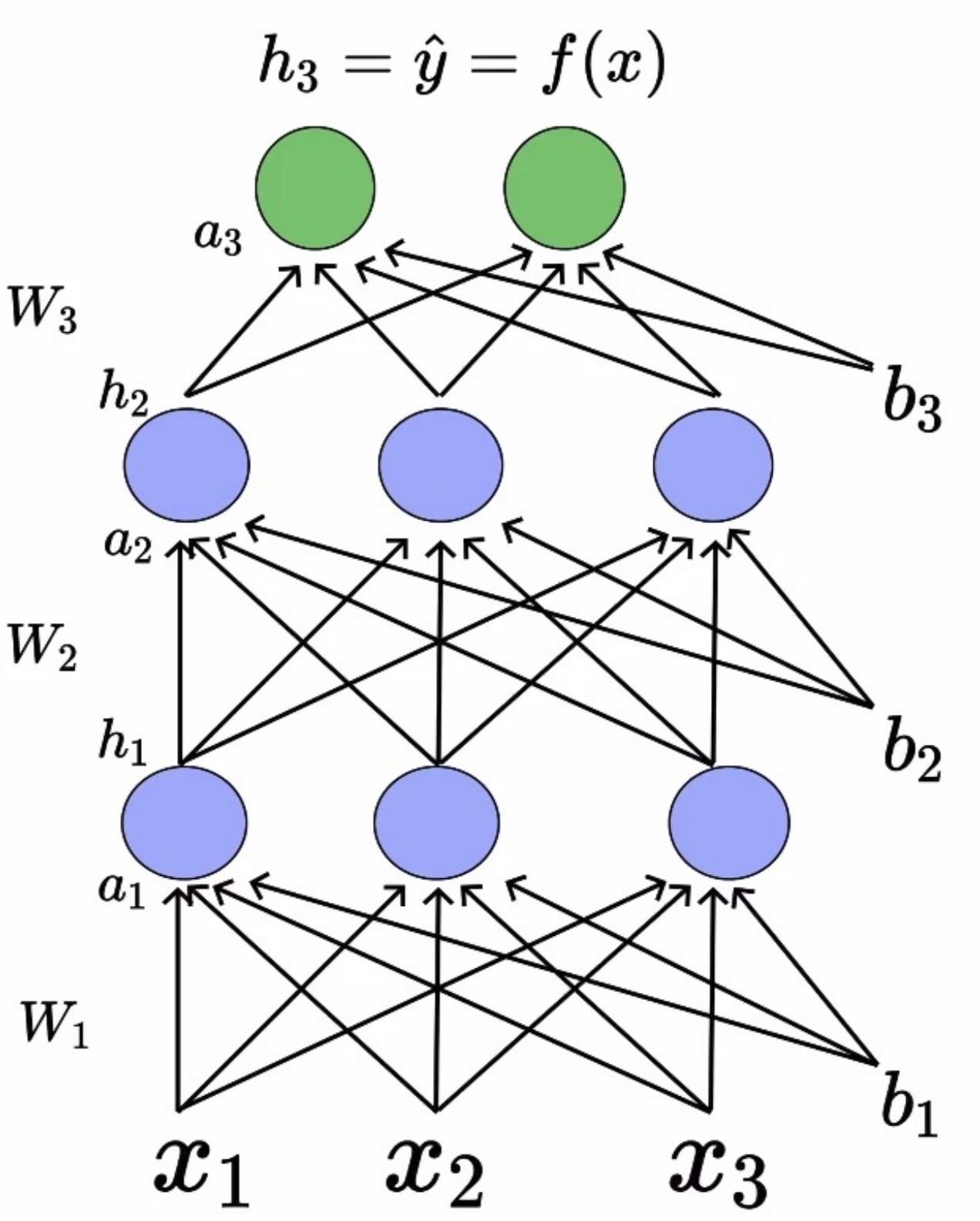


(Source:OneFourthLabs)

Here as we know the from already that, **a neuron takes all the inputs and weighted summation of all inputs and based on that the inputs.**

This aggregation is called the **pre-activation** (the summation part taking all inputs).

what ever it does after this is called **activation** (the sigmoid function).


$h_3 = \hat{y} = f(x)$
a_3
W_3
h_2
b_3
a_2
W_2
h_1
b_2
a_1
W_1
b_1
x_1
x_2
x_3

- The pre-activation at layer 'i' is given by

$$a_i(x) = W_i h_{i-1}(x) + b_i$$

- The activation at layer 'i' is given by

$$h_i(x) = g(a_i(x))$$

where 'g' is called as the activation function

- The activation at output layer 'L' is given by

$$f(x) = h_L = O(a_L)$$

where 'O' is called as the output activation function

Like that the neural network is built (Source: OneFourthLabs)

Lets consider particular deep neural network, has 10 layers and 100 neurons in each layer. with the corresponding weight matrices will be as below.

$$W_1 = \begin{bmatrix} w_{1\,1\,1} & w_{1\,1\,2} & . & . & . & w_{1\,1\,99} & w_{1\,1\,100} \\ w_{1\,2\,1} & w_{1\,2\,2} & . & . & . & w_{1\,2\,99} & w_{1\,2\,100} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ w_{1\,10\,1} & w_{1\,10\,2} & . & . & . & w_{1\,10\,99} & w_{1\,10\,100} \end{bmatrix} \qquad X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_{100} \end{bmatrix}$$

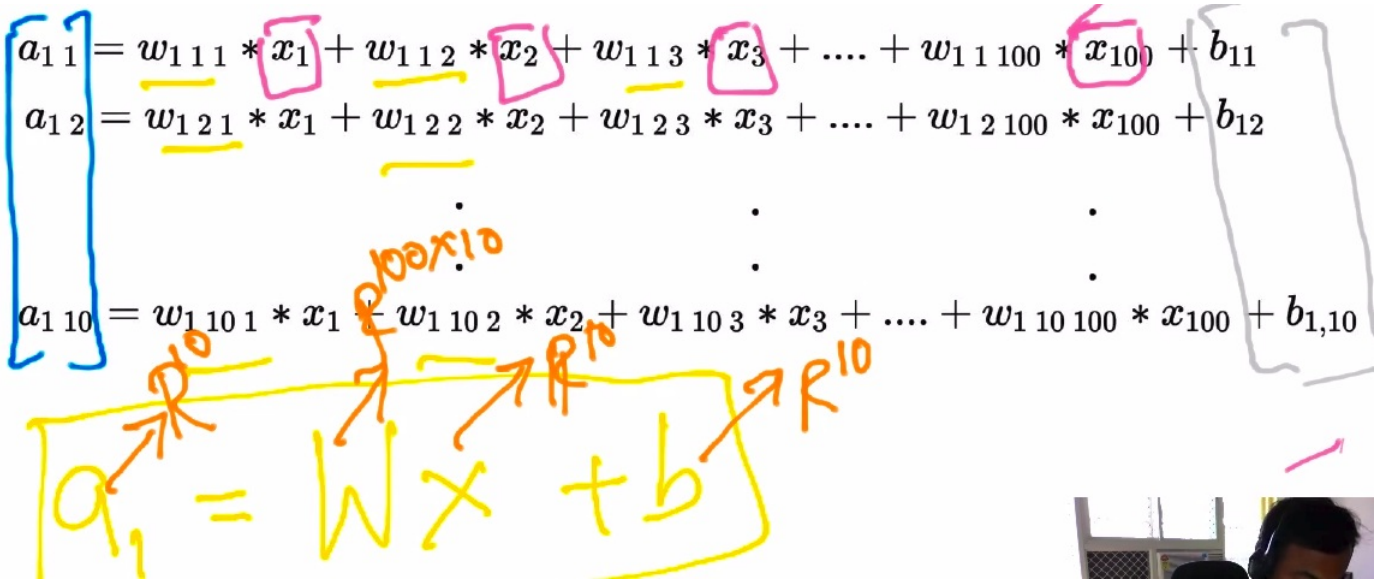
$$a_{1\,1} = w_{1\,1\,1} * x_1 + w_{1\,1\,2} * x_2 + w_{1\,1\,3} * x_3 + ... + w_{1\,1\,100} * x_{100} + b_{11}$$

$$a_{1\,2} = w_{1\,2\,1} * x_1 + w_{1\,2\,2} * x_2 + w_{1\,2\,3} * x_3 + ... + w_{1\,2\,100} * x_{100} + b_{12}$$

$$\begin{matrix} . & . & . \\ . & . & . \end{matrix}$$

$$a_{1\,10} = w_{1\,10\,1} * x_1 + w_{1\,10\,2} * x_2 + w_{1\,10\,3} * x_3 + ... + w_{1\,10\,100} * x_{100} + b_{1,10}$$

Weight matrices and each quantity value (Source: OneFourthLabs)

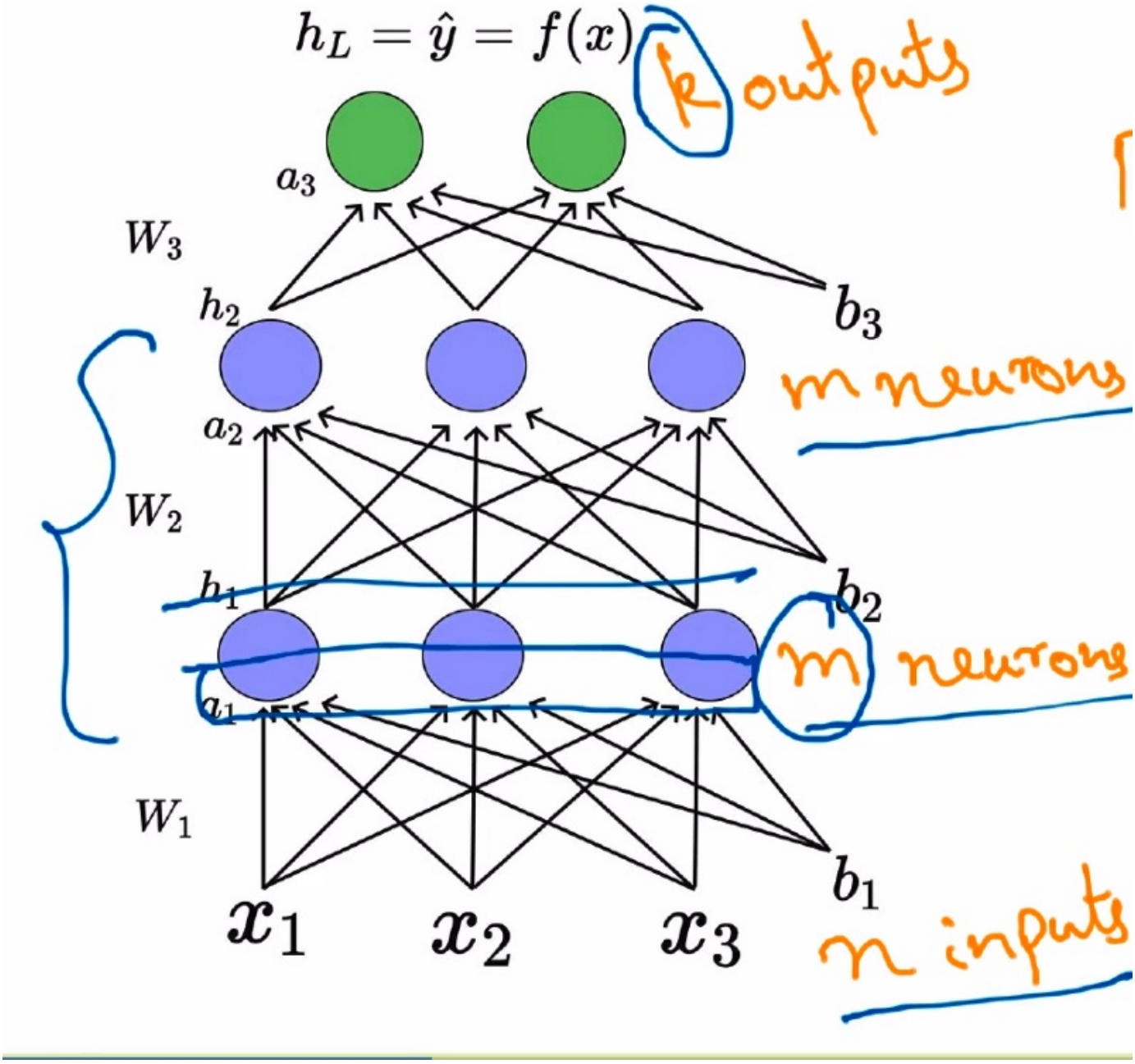


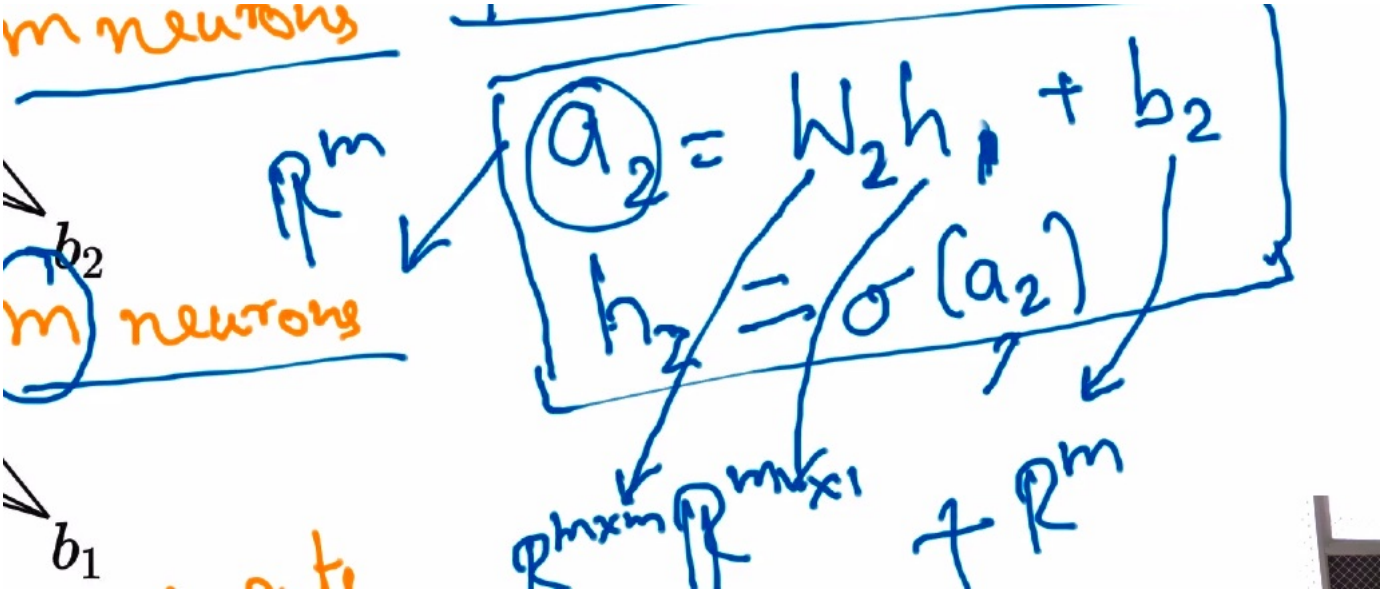
(Source:OneFourthLabs)

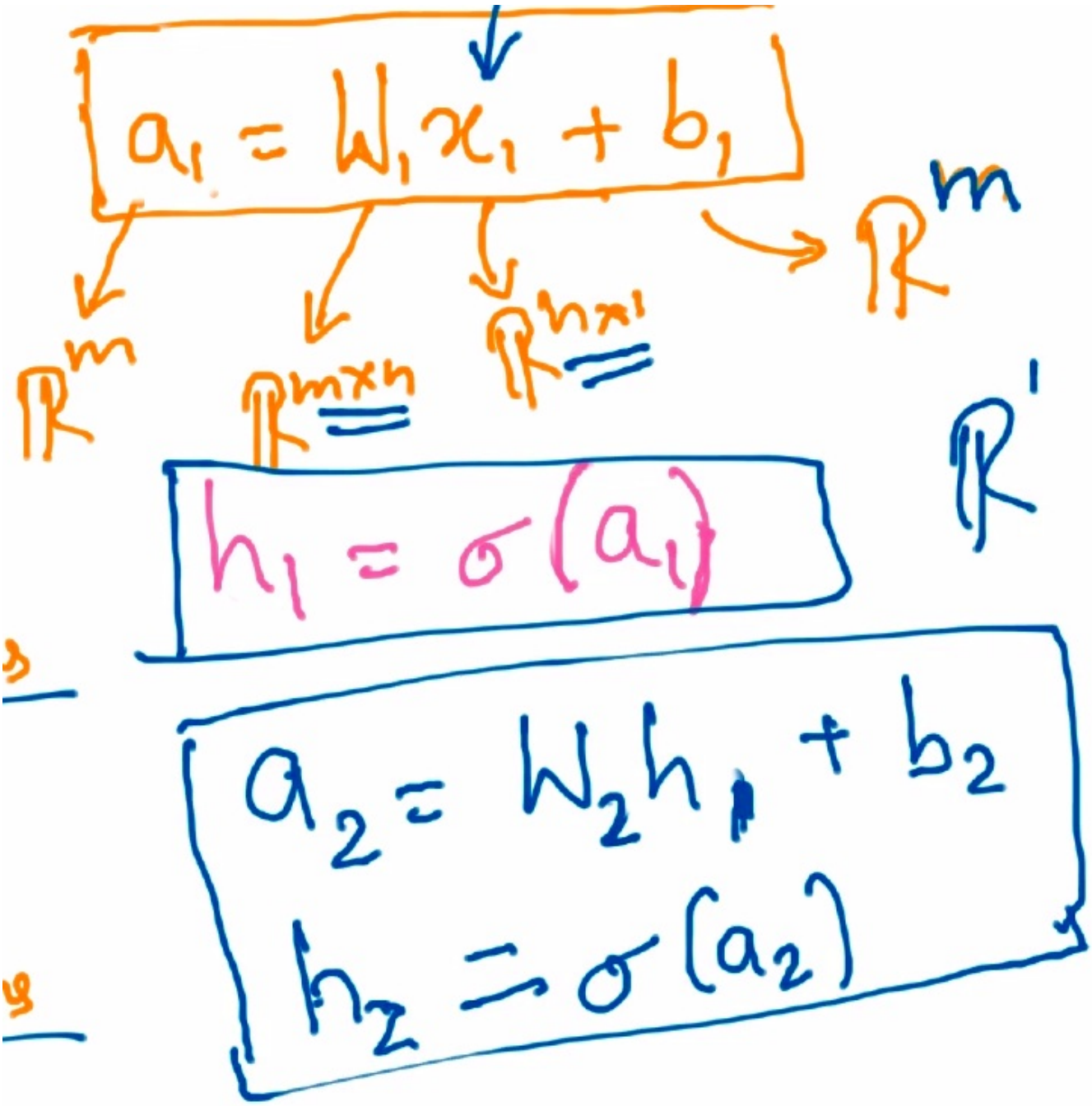
Here the weight matrix of dimension [100x100] and the the X matrix is of dimension [100x1].

Product of the two matrices will output a matrix of the dimensions (10X1) which is filled up with a11,a12,a13,..... ,a110.

Now, we will calculate the quantities of the second layer as shown in the below pictures.

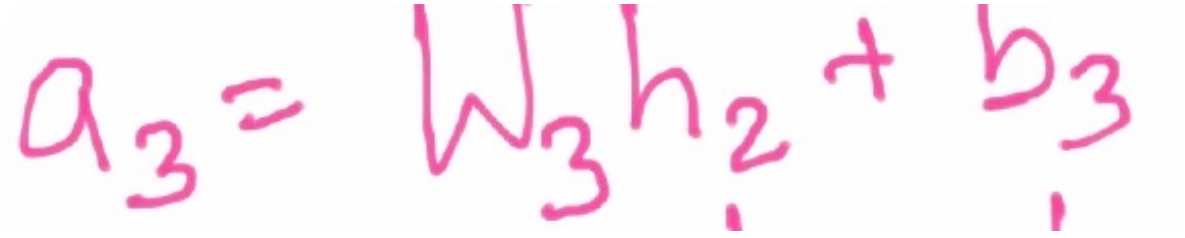

$h_L = \hat{y} = f(x)$
k outputs
a_3
W_3
h_2
b_3
m neurons
a_2
W_2
b_1
b_2
m neurons
a_1
W_1
b_1
x_1
x_2
x_3
n inputs


m neurons
$\mathbb{R}^m$
$a_2 = W_2 h_1 + b_2$
$h_2 = \sigma(a_2)$
b_2
m neurons
b_1
$\mathbb{R}^{m \times m}$ $\mathbb{R}^{m \times 1}$ + $\mathbb{R}^m$

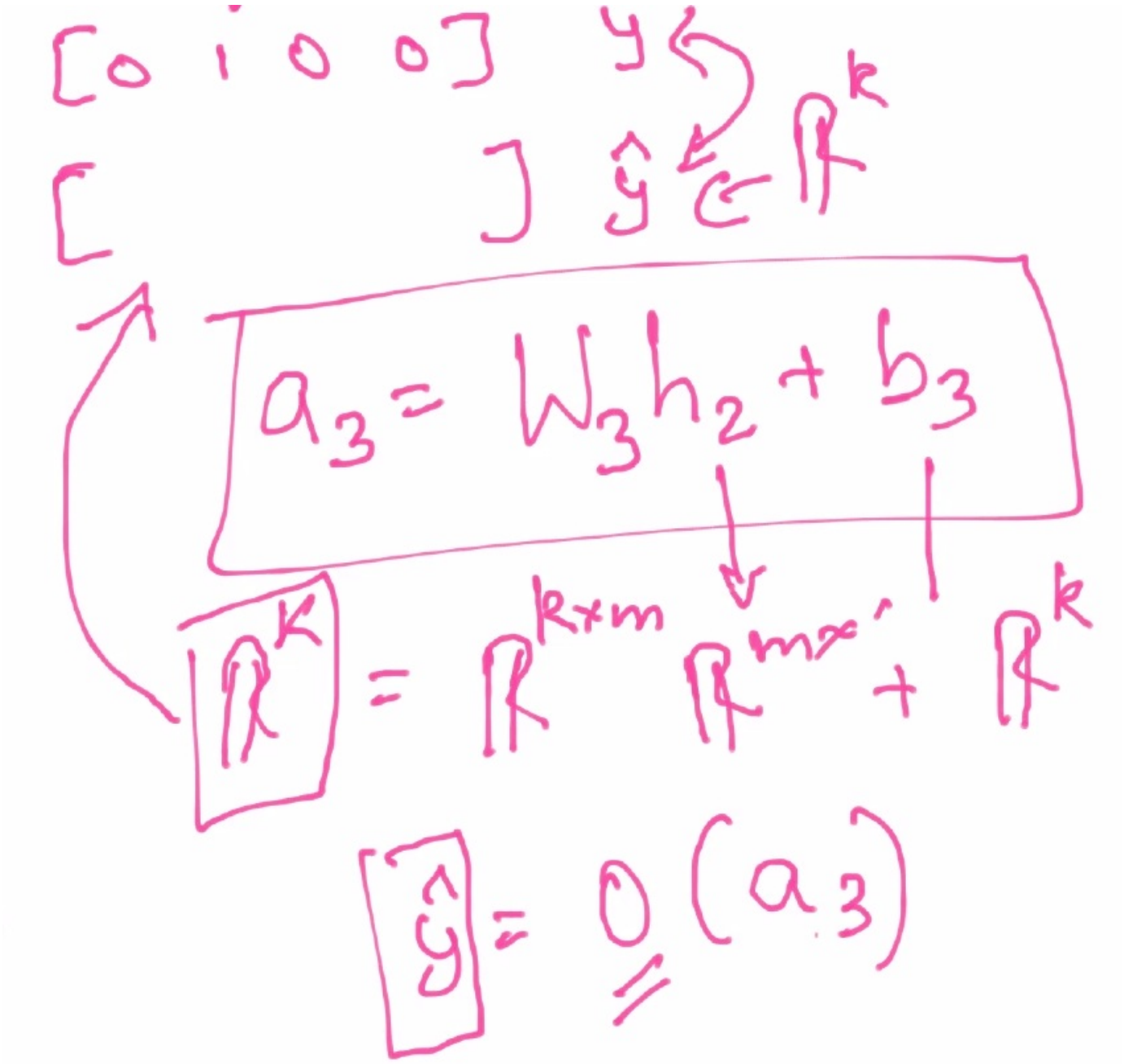


Like we will compute the quantities of second layer using the first layer values and applying sigmoid to those.(Source:OneFourthLabs)

Now upto h2 we have done the computation, we need compute it to the final top layer .



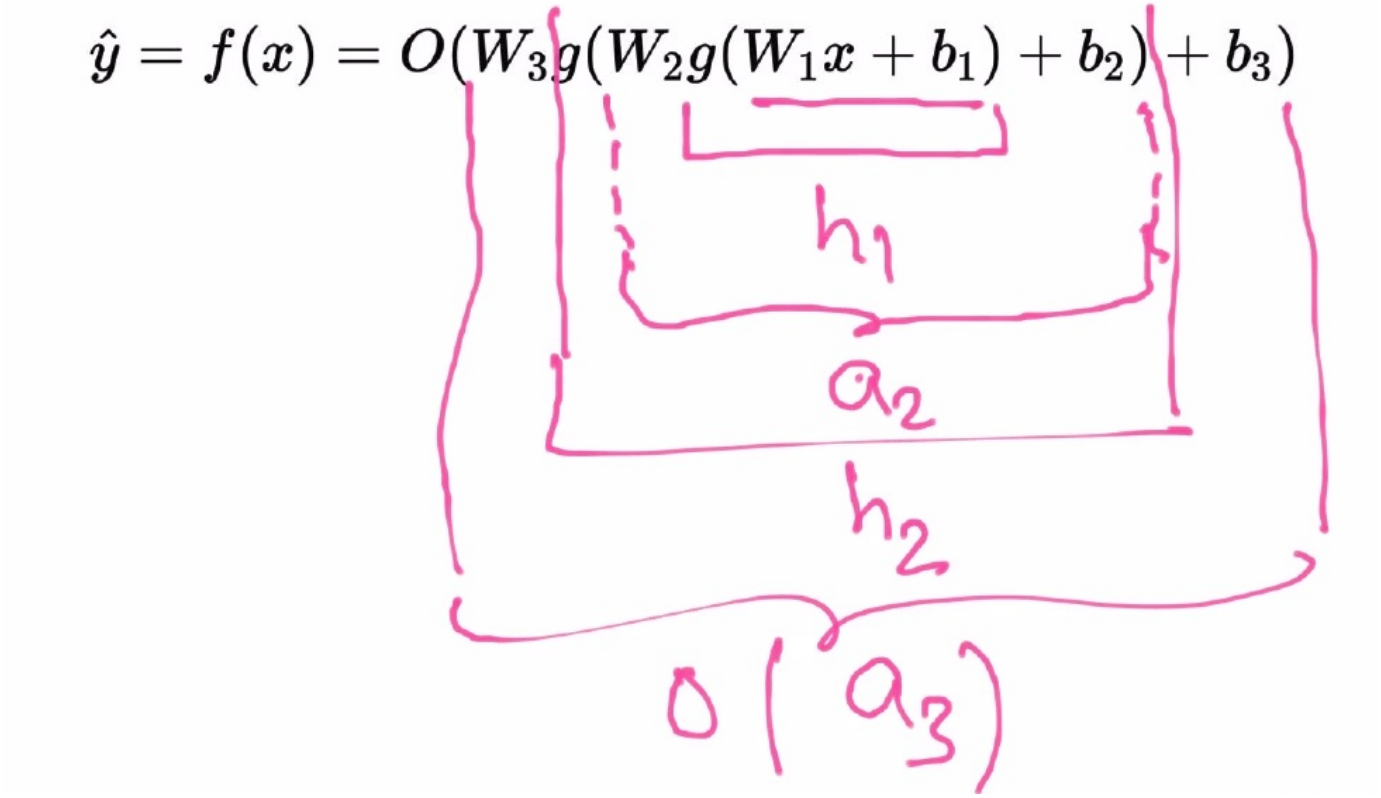
We know this from pervious compuataion (Source:OneFourthLabs).



the main doubt here is what is final O() function (Source: OneFourthLabs).

Here, like that finally we will get a3 of the dimensions of the [kx1] and we will lastly compute the final 'y hat' with O(a3). O() is the final function which we need.

$$\hat{y} = f(x) = O(W_3 g(W_2 g(W_1 x + b_1) + b_2) + b_3)$$



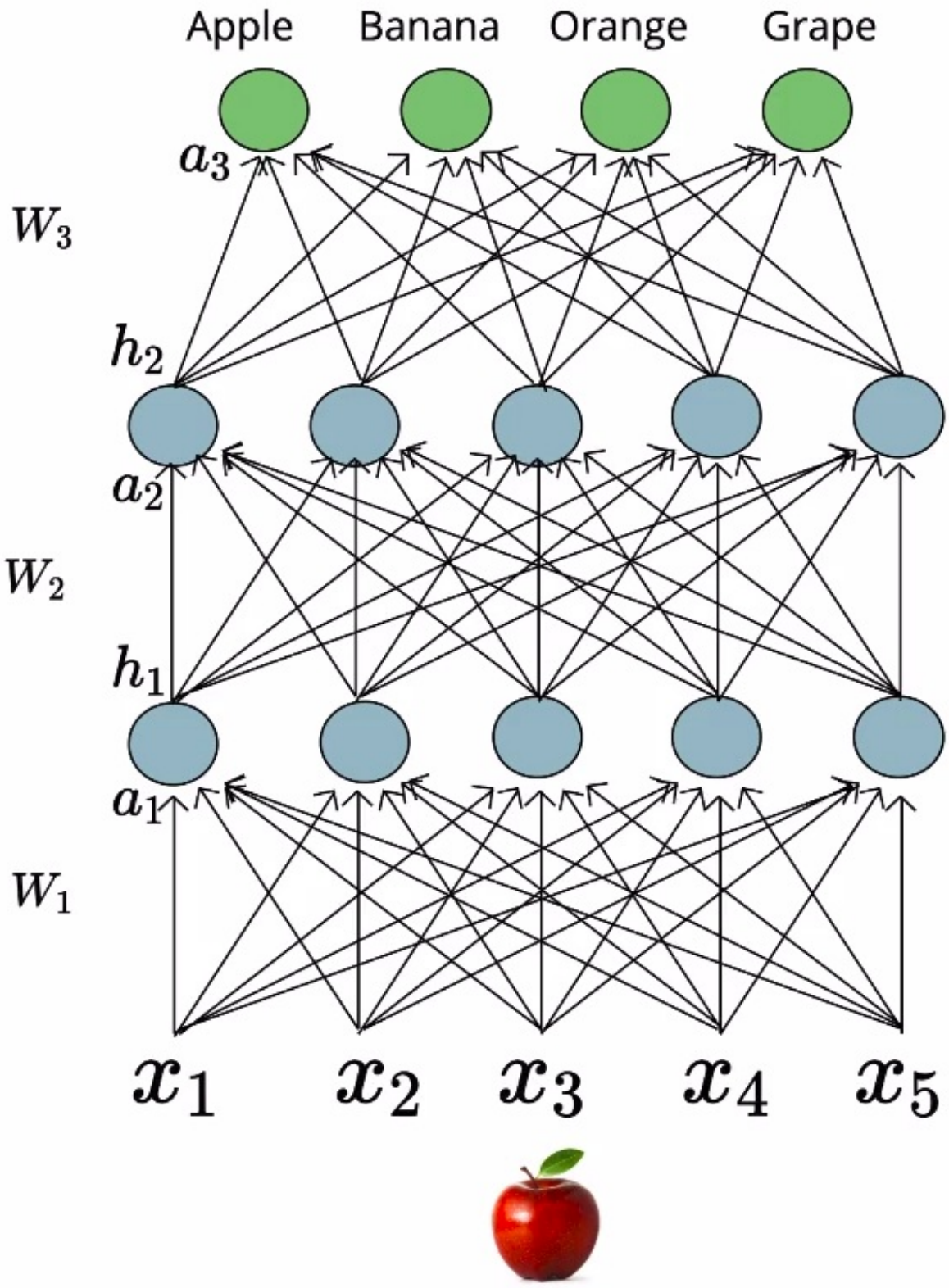
This the final 'y hat' from the previous computation (Source:OneFourthLabs).

Finally if you take all the dimensions ad see the output dimensions, that vector will be in the dimensions of K.

We can validate the function O( ) in two ways:

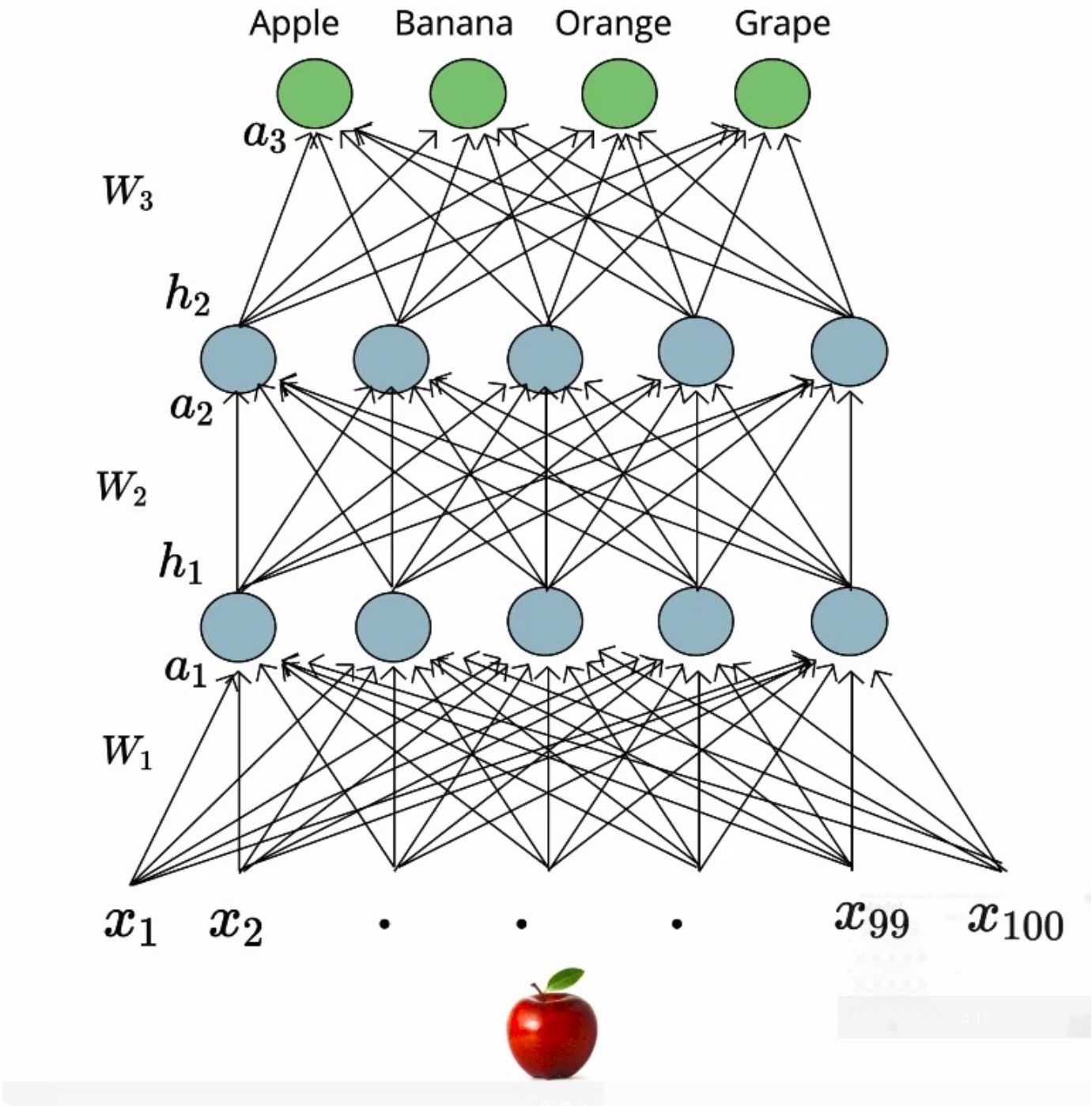
1. Classification
2. Regression

Classification:



(Source:OneFourthLabs).

Here, we need to take particular inputs of a fruit and we need to pass through various hidden layers and we need to outputs which fruit is that.This comes under multi class classification because the output will be one among the several fruits.


Apple
Banana
Orange
Grape
a_3
W_3
h_2
a_2
W_2
h_1
a_1
W_1
x_1
x_2
x_{99}
x_{100}

$$W_3 = \begin{bmatrix} w_{3\,1\,1} & w_{3\,1\,2} & . & . & . & w_{3\,1\,10} \\ w_{3\,2\,1} & w_{3\,2\,2} & . & . & . & w_{3\,2\,10} \\ w_{3\,3\,1} & w_{3\,3\,2} & . & . & . & w_{3\,3\,10} \\ w_{3\,4\,1} & w_{3\,4\,2} & . & . & . & w_{3\,4\,10} \end{bmatrix} \qquad h_2 = \begin{bmatrix} h_{2\,1} \\ h_{2\,2} \\ . \\ . \\ h_{2\,10} \end{bmatrix}$$

$$a_{3\,1} = w_{3\,1\,1} * h_{2\,1} + w_{3\,1\,2} * h_{2\,2} + w_{3\,1\,3} * h_{2\,3} + ... + w_{3\,1\,10} * h_{2\,10} + b_{31}$$

$$a_{3\,2} = w_{3\,2\,1} * h_{2\,1} + w_{3\,2\,2} * h_{2\,2} + w_{3\,2\,3} * h_{2\,3} + ... + w_{3\,2\,10} * h_{2\,10} + b_{32}$$

$$a_{3\,3} = w_{3\,3\,1} * h_{2\,1} + w_{3\,3\,2} * h_{2\,2} + w_{3\,3\,3} * h_{2\,3} + ... + w_{3\,3\,10} * h_{2\,10} + b_{33}$$

$$a_{3\,4} = w_{3\,4\,1} * h_{2\,1} + w_{3\,4\,2} * h_{2\,2} + w_{3\,4\,3} * h_{2\,3} + ... + w_{3\,4\,10} * h_{2\,10} + b_{34}$$

$$a_3 = W_3 * h_2 + b_3$$

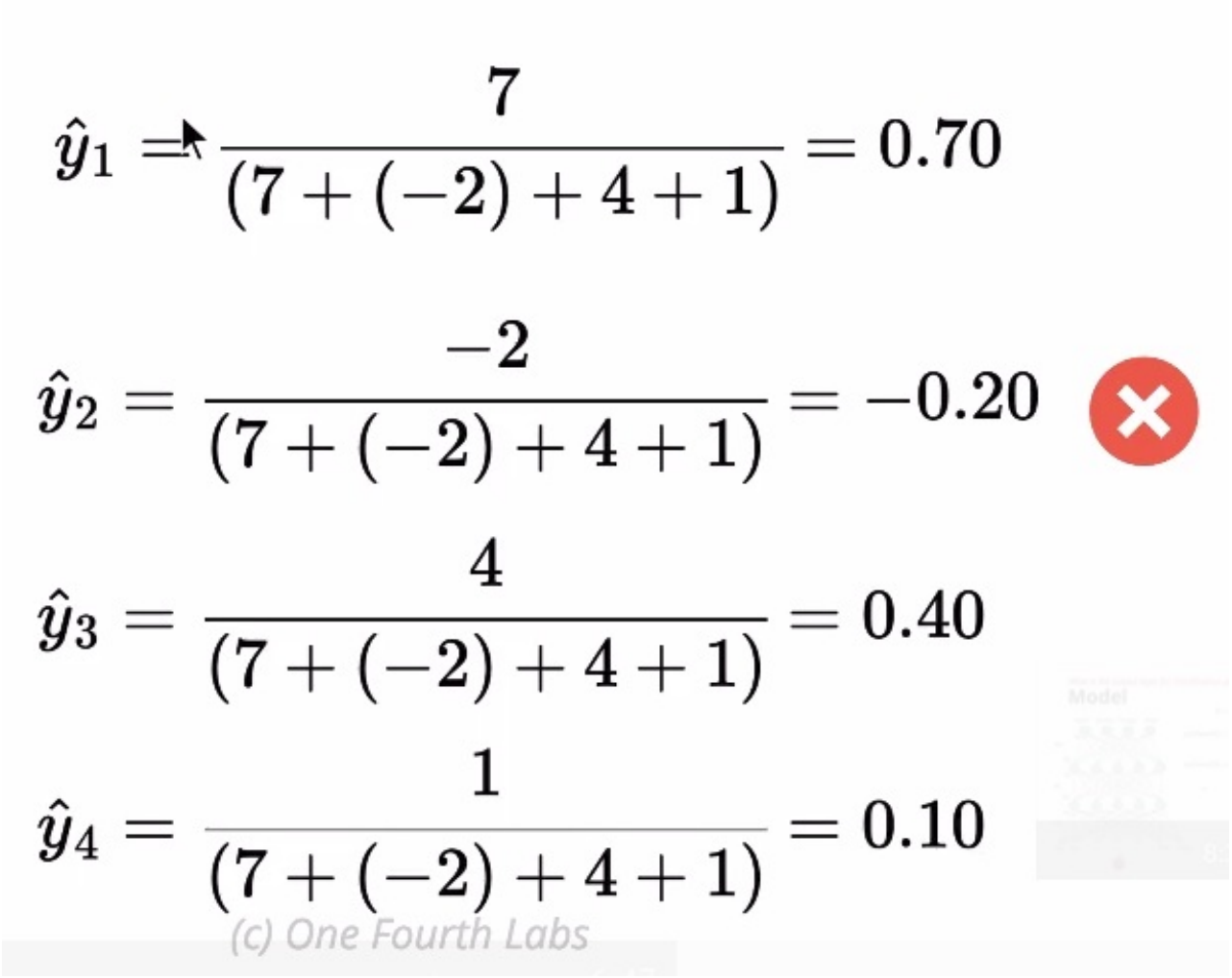
For this deep neural network compute a3 using various computations (Source: OneFourthLabs).

$$Say\ for\ other\ input\ a_3 = [\ 7 \quad -2 \quad 4 \quad 1\ ]$$

Output Activation Function has to be chosen such that output is probability

(Source:OneFourthLabs).

If we give the input a3 as the particular values the output activation function has to be chosen such that output is probability.



(Source:OneFourthLabs).

We will compute each y hat like “y1 hat”, “y2 hat” , “y3 hat” and …. w we will see the probability of each as show.

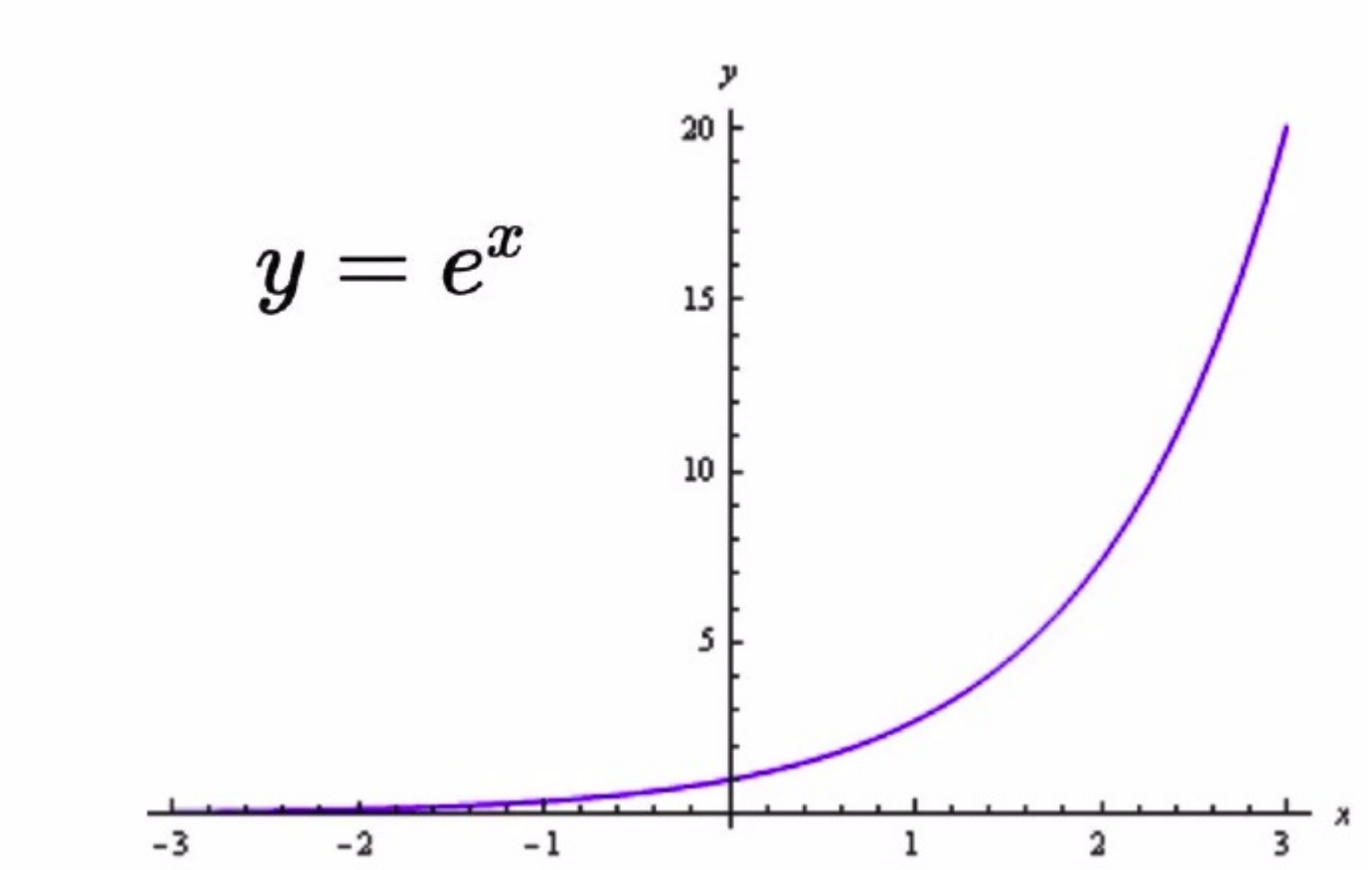
Here this will not work in all the cases quantities will be negative as shown in the beside image.

In this case we will introduce the softmax function to overcome this drawback a well known function in deep learning.

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad for\ i = 1.....k$$

Softmax Function (Source:OneFourthLabs).

We use this because because we softmax function has a separate quality.Lets look at the graph of that.



Graph of e^x (Source:OneFourthLabs).

The function e^x is outputs positive value even fr negative inputs that's why we use the to get positive output even there is a negative output.Based on that we will define the softmax function.

So, we will apply the softmax function to the h as show below and compute the values follows.

$$h = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 \end{bmatrix}$$

$$softmax(h) = [softmax(h_1) \quad softmax(h_2) \quad softmax(h_3) \quad softmax(h_4)]$$

$$softmax(h) = \left[\frac{e^{h_1}}{\sum_{j=1}^{4} e^{h_j}} \quad \frac{e^{h_2}}{\sum_{j=1}^{4} e^{h_j}} \quad \frac{e^{h_3}}{\sum_{j=1}^{4} e^{h_j}} \quad \frac{e^{h_4}}{\sum_{j=1}^{4} e^{h_j}} \right]$$

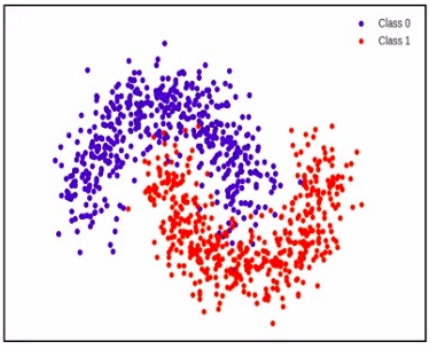
$$softmax(h_i) \; is \; the \; i^{th} \; element \; of \; softmax \; output$$

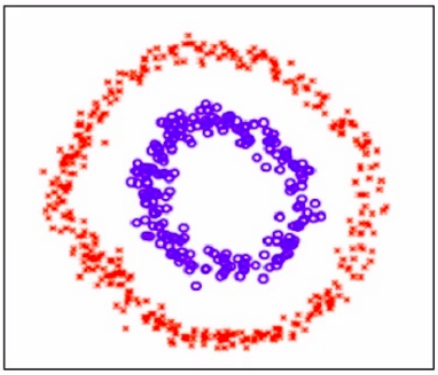
(Source:OneFourthLabs).

Finally we apply **sigmoid** for the **first layer and second layer** and finally for the top most layer i.e., the **output layer** we apply the **softmax function.**

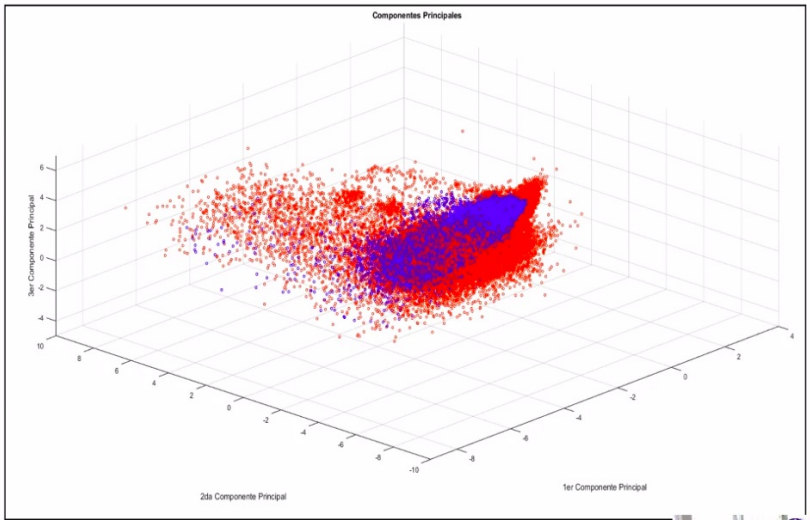
Now if we com back to the examples if we come back to the non linearly separable data id we have 2 inputs ad 3 inputs.

x_1	x_2
-	-
-	-
-	-





x_1	x_2	x_3
-	-	-
-	-	-
-	-	-



if we have two inputs ans 3 inputs data-sets can be like this (Source: OneFourthLabs).

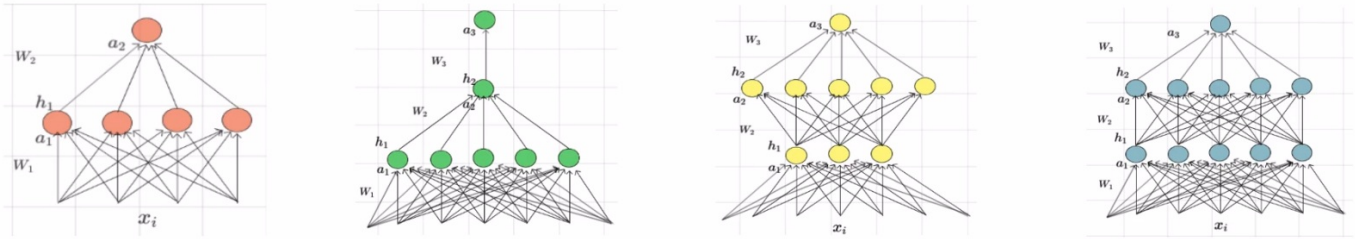
x_1	x_2	x_3	x_4
-	-	-	-
-	-	-	-
-	-	-	-

(Source:OneFourthLabs).

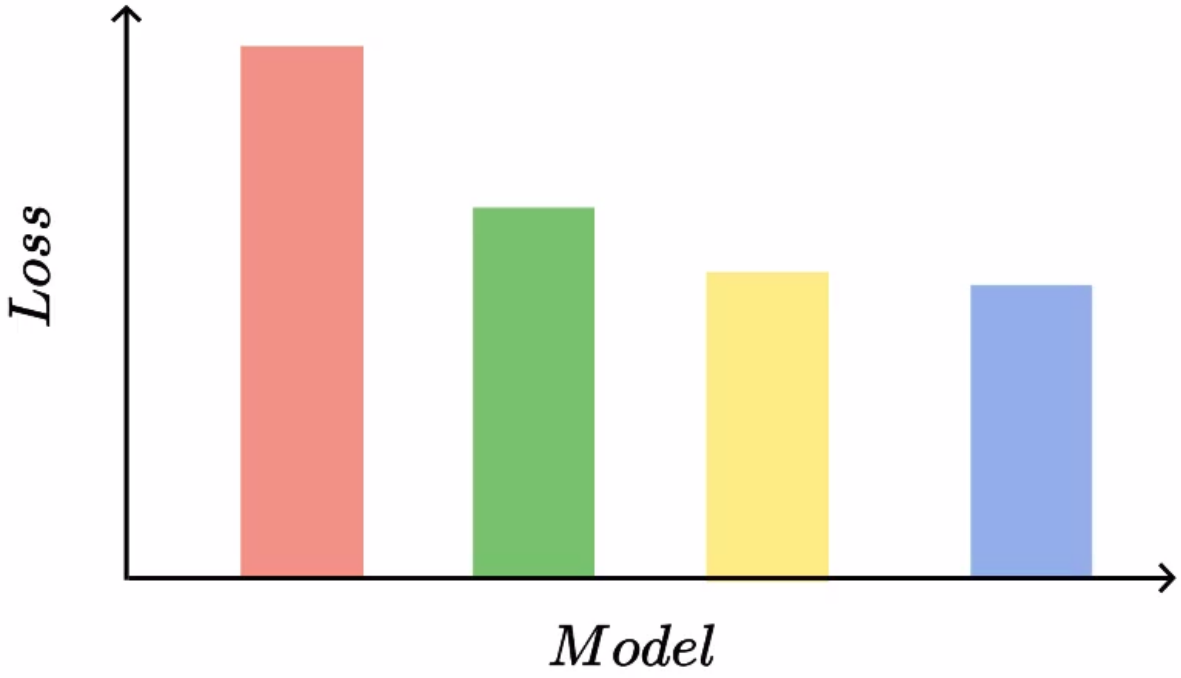
What if we have 4 inputs ? like 4 dimensions, in most of cases in real world data we will be dealing with 100, 1000 inputs of the data where there is no chance for us to realise how the data will be.

Beyond 4 dimensions we cant visualise how the data will be.

Then we will try a deep neural network with different parameter tuning.With one hidden layer and one neuron in the output layer, two hidden layers with 5 neurons in one and 1 neuron in another and another neural networks like that.



Try different deep neural networks as described above (Source: OneFourthLabs).

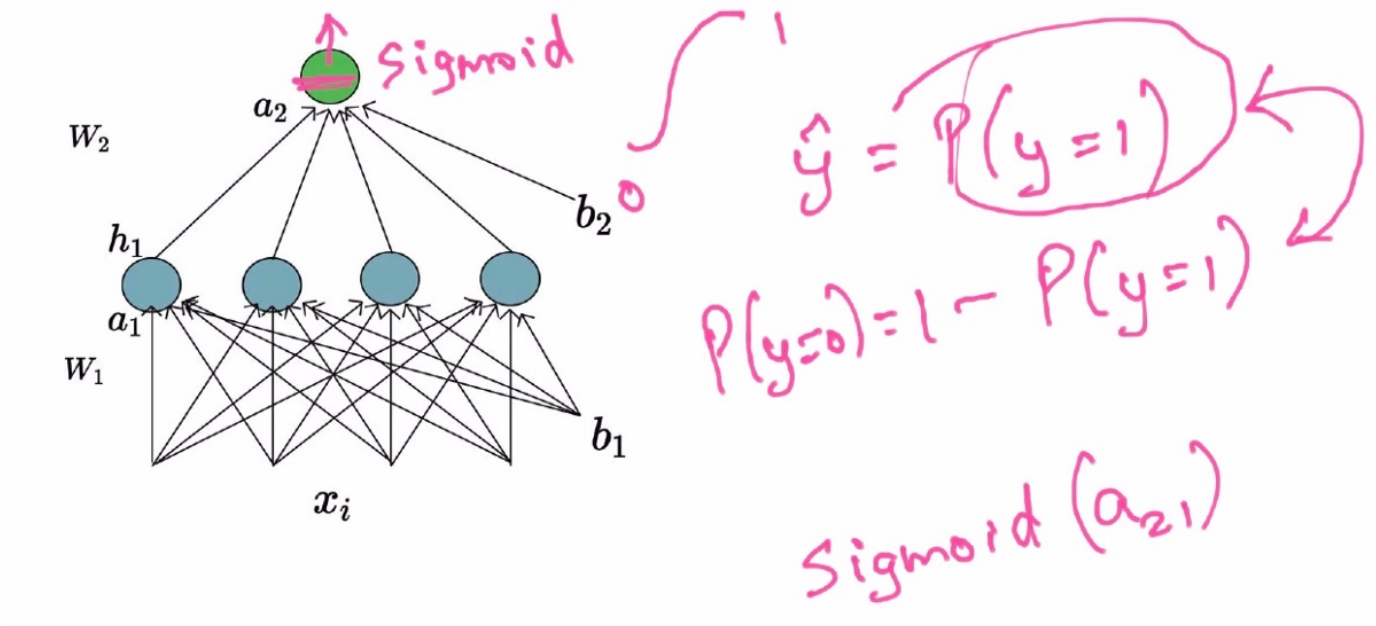


(Source:OneFourthLabs).

Like these we are going to try different neural network for each model and then we will compute the loss of the all functions.Looking at the loss we will choose the model this is what called **hyper parameter tuning**.

Loss Function:

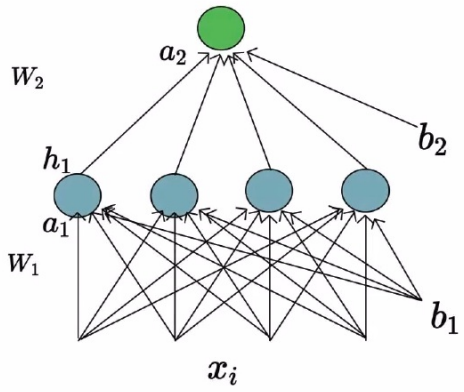
For binary class classification:



(Source:OneFourthLabs).

In case of the binary class classification we will have only one neuron at the end finally we only need to apply the sigmoid function at then end.

Here in this case for comfort sake lets assume particular values of input and true output. we will apply sigmoid function as usual and we will get the predicted y values as after following all the computational steps the we will use the cross entropy Loss function. and compute the loss as shown.



$$b = \begin{bmatrix} 0.5 & 0.3 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} 0.9 & 0.2 & 0.4 & 0.3 \\ -0.5 & 0.4 & 0.3 & 0.3 \\ 0.1 & 0.1 & -0.1 & 0.2 \\ -0.2 & 0.5 & 0.5 & 0.7 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.5 & 0.8 & -0.6 & 0.3 \end{bmatrix}$$

$$x = \begin{bmatrix} 0.3 & 0.5 & -0.4 & 0.3 \end{bmatrix} \qquad y = 1$$

Output :

$$a_1 = W_1 * x + b_1 = \begin{bmatrix} 0.8 & 0.52 & 0.68 & 0.7 \end{bmatrix}$$

$$h_1 = sigmoid(a_1) = \begin{bmatrix} 0.69 & 0.63 & 0.66 & 0.67 \end{bmatrix}$$

$$a_2 = W_2 * h_1 + b_2 = 0.948$$

$$\hat{y} = sigmoid(a_2) = 0.7207$$

Cross Entropy Loss:

$$L(\Theta) = \begin{cases} -log(\hat{y}) & \text{if } y = 1 \\ -log(1-\hat{y}) & \text{if } y = 0 \end{cases}$$

$$L(\Theta) = -1 * \log(0.7207)$$

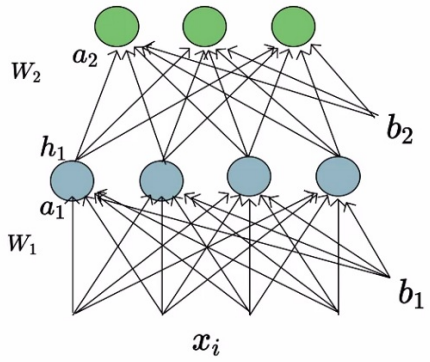
$$= 0.327$$

How we compute loss for binary type of classification (Source:OneFourthLabs).

For Multi class classification:

Here there will be multiple labels and out out which one turns out true in the output

Here we have two layers so we will apply sigmoid first the first layer and as the second layer is the top most layer we will use softmax and get the predicted ouptu value.



$$x = [\ 0.2 \quad 0.5 \quad -0.3 \quad 0.3\ ] \qquad y = [\ 0 \quad 1 \quad 0\ ]$$

Output :

$$\begin{aligned} a_1 &= W_1 * x + b_1 = [\ -0.19 \quad -0.16 \quad -0.09 \quad 0.77\ ] \\ h_1 &= sigmoid(a_1) = [\ 0.45 \quad 0.46 \quad 0.49 \quad 0.68\ ] \\ a_2 &= W_2 * h_1 + b_2 = [\ 0.13 \quad 0.33 \quad 0.89\ ] \\ \hat{y} &= softmax(a_2) = [\ 0.23 \quad 0.28 \quad 0.49\ ] \end{aligned}$$

$$b = [\ 0 \quad 0\ ]$$

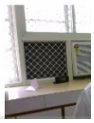
$$W_1 = \begin{bmatrix} 0.1 & 0.3 & 0.8 & -0.4 \\ -0.3 & -0.2 & 0.5 & 0.5 \\ -0.3 & 0.1 & 0.5 & 0.4 \\ 0.2 & 0.5 & -0.9 & 0.7 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.3 & 0.8 & -0.2 & -0.4 \\ 0.5 & -0.2 & -0.3 & 0.5 \\ 0.3 & 0.1 & 0.6 & 0.6 \end{bmatrix}$$

Cross Entropy Loss:

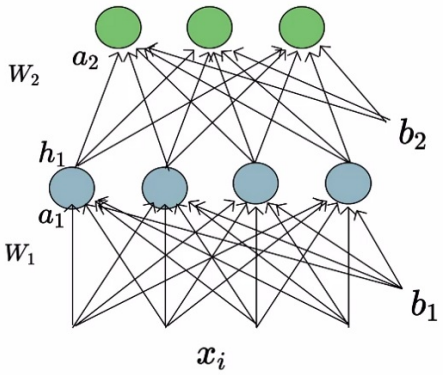
$$L(\Theta) = -\sum_{i=1}^{k} y_i \log(\hat{y}_i)$$

$$\begin{aligned} L(\Theta) &= -1 * \log(0.28) \\ &= 1.2729 \end{aligned}$$



Here loss is very high as we assigned lot of probability mass of 3rd element in softmax function id its appropriate to have a high loss (Source:OneFourthLabs).

If we take other values and try doing all the stuff again same as the earlier.



$$x = [\ 0.3 \quad -0.4 \quad 0.6 \quad 0.2\ ] \quad y = [\ 0 \quad 0 \quad 1\ ]$$

Output :

$$\begin{aligned} a_1 &= W_1 * x + b_1 = [\ 0.31 \quad 0.39 \quad 0.25 \quad -0.54\ ] \\ h_1 &= sigmoid(a_1) = [\ 0.58 \quad 0.60 \quad 0.56 \quad 0.37\ ] \\ a_2 &= W_2 * h_1 + b_2 = [\ 0.39 \quad 0.18 \quad 0.79\ ] \\ \hat{y} &= softmax(a_2) = [\ 0.3024 \quad 0.2462 \quad 0.4514\ ] \end{aligned}$$

$$b = [\ 0 \quad 0\ ]$$

$$W_1 = \begin{bmatrix} 0.1 & 0.3 & 0.8 & -0.4 \\ -0.3 & -0.2 & 0.5 & 0.5 \\ -0.3 & 0.1 & 0.5 & 0.4 \\ 0.2 & 0.5 & -0.9 & 0.7 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.3 & 0.8 & -0.2 & -0.4 \\ 0.5 & -0.2 & -0.3 & 0.5 \\ 0.3 & 0.1 & 0.6 & 0.6 \end{bmatrix}$$

Cross Entropy Loss:

$$L(\Theta) = -\sum_{i=1}^{k} y_i \log(\hat{y}_i)$$

$$\begin{aligned} L(\Theta) &= -1 * \log(0.4514) \\ &= 0.7954 \end{aligned}$$

(Source:OneFourthLabs)

Learning Algorithm:

we know basic learning algorithm from before as below.

Initialise w, b

Iterate over data:

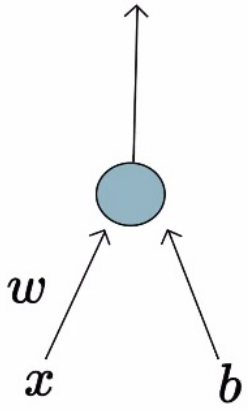
$compute\ \hat{y}$

$compute\ \mathscr{L}(w, b)$

$w = w - \eta \Delta w$

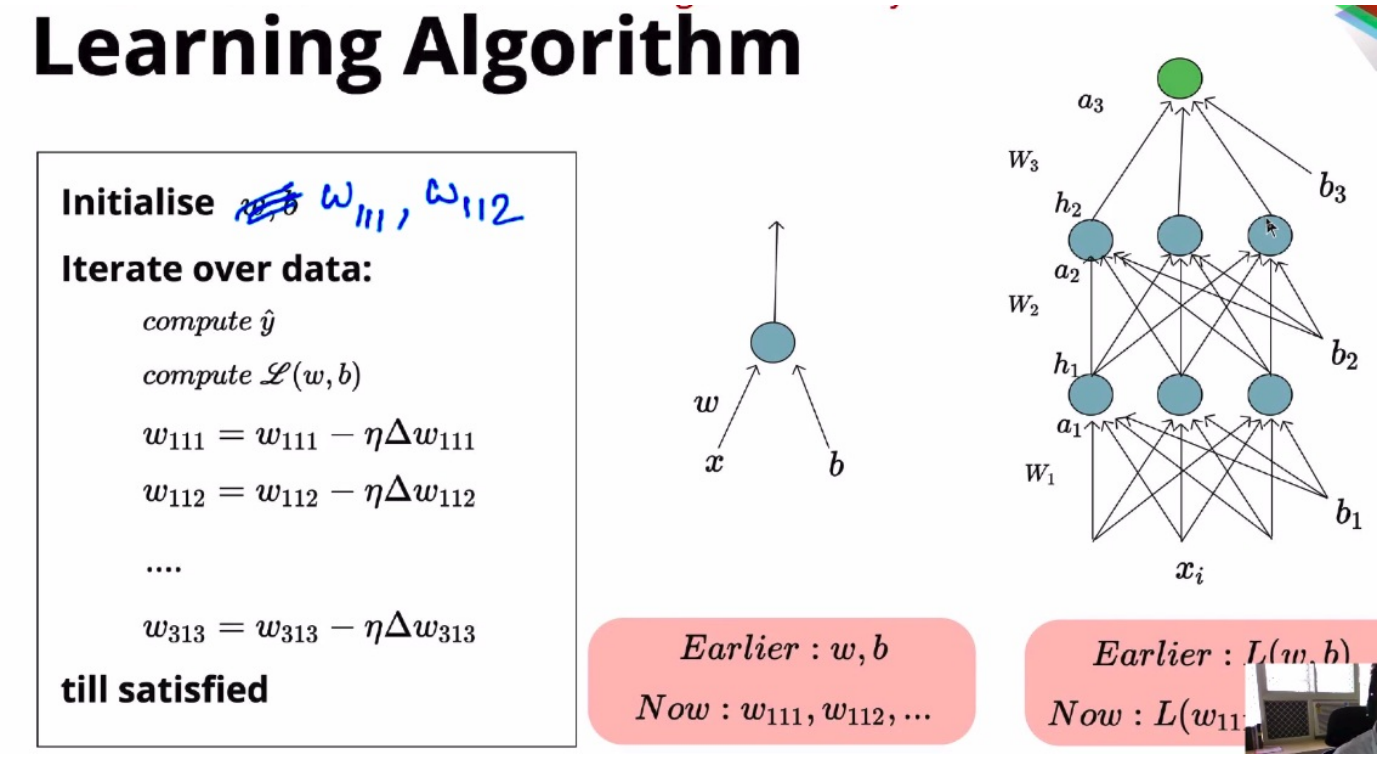
$b = b - \eta \Delta b$

till satisfied



(Source:OneFourthLabs)

The change we have we seen in the below image.



(Source:OneFourthLabs)

Evaluation:

Binary Class Classification:

Given the dataset of the patient we need to take varous parameters called Age,Albumin,T_Bilirium,..... and we need to predict whether the patient needs to be diagnosed or not . Based on the predictions we will calculate the accuracy of the model as shown below.

Indian Liver Patient Records * - whether person needs to be diagnosed or not ?

Test Data

Age	Albumin	T_Bilirubin
65	3.3	0.7
62	3.2	10.9
20	4	1.1
84	3.2	0.7

...

y	Predicted	
0	0	✓
0	1	✗
1	1	✓
1	0	✗

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

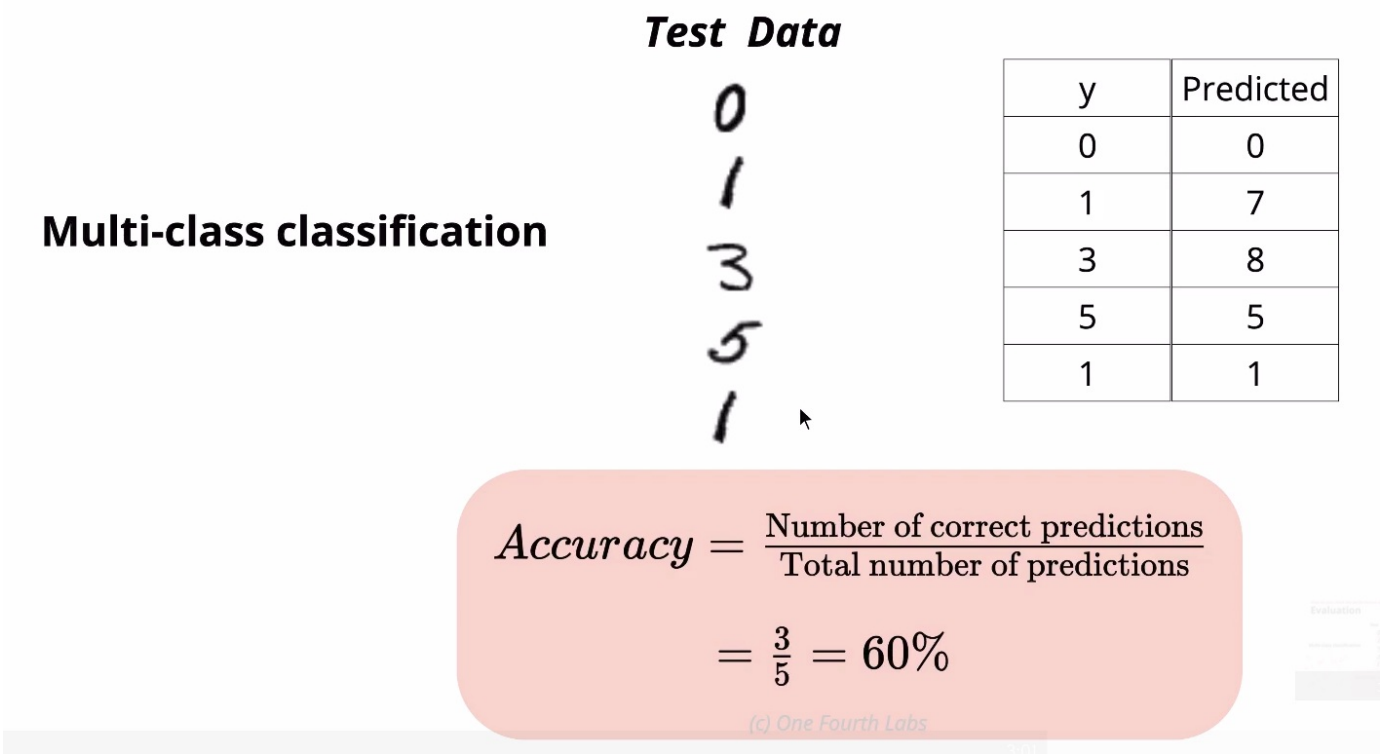
$$= \frac{2}{4} = 50\%$$

(c) One Fourth Labs

(Source:OneFourthLabs).

Multi-Class Classification:

Given the data-set of the MNIST and we need to predict the what number it is this comes under the multi class classification.



(Source:OneFourthLabs)

This is about the feed forward neural networks use to classify non linearly separable data using complex functions with help of building blocks called sigmoid neurons.

This is a small try ,uploading the notes . I believe in "**Sharing knowledge is that best way of developing skills**".Comments will be appreciated. Even small edits can be suggested.

> **Each Applause will be a great encouragement.**

Do follow my medium for more updates.....